USING MACHINE LEARNING METHODS FOR ANALYZING SEQUENTIAL TEXTUAL DATA

Erik Novak

Doctoral Dissertation
Jožef Stefan International Postgraduate School
Ljubljana, Slovenia

Supervisor: Prof. Dr. Dunja Mladenić, IPS and Jožef Stefan Institute, Ljubljana, Slovenia

Evaluation Board:
Prof. Dr. Tomaž Erjavec, Chair, IPS and Jožef Stefan Institute, Ljubljana, Slovenia
Prof. Dr. Jasminka Dobša, Member, Faculty of Organization and Informatics, University of Zagreb, Varaždin, Croatia
Prof. Dr. Primož Škraba, Member, Jožef Stefan Institute, Ljubljana, Slovenia and Queen Mary University of London, London, United Kingdom

MEDNARODNA PODIPLOMSKA ŠOLA JOŽEFA STEFANA
JOŽEF STEFAN INTERNATIONAL POSTGRADUATE SCHOOL

Erik Novak

USING MACHINE LEARNING METHODS FOR ANALYZING SEQUENTIAL TEXTUAL DATA

Doctoral Dissertation

UPORABA METOD STROJNEGA UČENJA ZA ANALIZIRANJE TEKSTOVNIH PODATKOV V ZAPOREDJU

Doktorska disertacija

Supervisor: Prof. Dr. Dunja Mladenić

Ljubljana, Slovenia, September 2024

To my family and friends

Acknowledgments

This thesis would not have been finished without the support of many people who have helped me on this journey. First, I would like to thank my supervisor, Dunja Mladenić, who encouraged me to pursue a PhD and a career in academia and has guided me through my research throughout these years. Big thanks also to the members of the thesis evaluation board, Tomaž Erjavec, Jasminka Dobša, and Primož Škraba, for providing helpful comments.

I would also like to thank my current and past colleagues from the Department for Artificial Intelligence, and other departments, at the Jožef Stefan Institute, especially Klemen Kenda, Luka Bizjak, Inna Novalija, M. Besher Massri, Marko Grobelnik, Jasna Franko, Mojca Kregar Zavrl, Mateja Škraba, Polona Škraba, Alenka Guček, Patrik Zajec, Kaja Dobrovoljc, Simon Krek, Gregor Leban, Jakob Jelenčič, Erik Calcina, Aljaž Košmerlj, Miha Torkar, Jasna Urbančič, Zala Herga Rupnik, Blaž Fortuna, Jan Rupnik, Andrej Muhič, Blaž Bertalanič, Gregor Cerar, and Carolina Fortuna, with whom I discussed research topics, bounced ideas, who helped me with administration work and are in general nice people to work with.

I want to thank all of my friends and family, especially my parents, Andrej and Andreja, my sister Mateja and my grandparents, Andrej and Ivanka. Your moral support, guidance, and discussions made this journey easier and helped me through the more difficult times. Furthermore, your frequent questions like "How is your PhD going?" and "When can we expect it to be finished?" motivated me to work harder and finish it.

Finally, I would like to thank Anamarija Kuhar, whom I am blessed to have in my life. You have been my strongest supporter, being there when I needed you the most.

The research in this thesis was funded by the Slovenian Research Agency, the following European Union Horizon 2020 projects: X5GON [Grant No. 761758], HumanE-AI-Net [Grant No. 952026], Water4Cities [Grant No. 734409], PerceptiveSentinel [Grant No. 776115], INFINITECH [Grant No. 856632], and the following European Union Horizon Europe projects: enRichMyData [Grant No. 101070284], PREPARE [Grant No. 101080288].

Abstract

This thesis investigates machine learning methods for analyzing sequential textual data. Sequential textual data refers to text collected in a specific order where the sequence is significant. Examples include (1) sentences, where word usage and order must adhere to the rules of grammar, (2) news reporting on events happening at different times, and (3) texts related to financial stock prices, reporting on the financial market dynamics. Analyzing this data helps us understand patterns, make predictions, and enhance decision-making for various tasks.

We first focus on sentence similarity using sentence structure, where we propose two methods, Language Model Earth Mover's Distance (LM-EMD) and Order-Preserving Wasserstein Score (OPWScore), that consider the word's contextual meaning and position in the sentence. The LM-EMD model was developed for cross-lingual information retrieval and measures the relevance of a document to a given query. The OPWScore metric evaluates text generation approaches, such as machine translation models. Both methods use language models to create the word's contextual embedding, which was then utilized to measure the texts' similarity using optimal transport, specifically the Wasserstein distance. OPWScore also restricts its distance calculation to words in similar positions, emphasizing the role of word placement. The methods are evaluated and compared with other models and metrics that consider sentence structure. The results show that sentence structure contributes to fluency-based performance while preserving the adequacy-related performance of the models.

Next, we explore the use of neural network models for online news clustering, considering the news' publication time. We first introduce a methodology for creating novel news clustering data sets. It automates the news collection and clustering process, thus reducing the time and resources required for evaluators to annotate the data sets manually. The methodology is used to create a cross-lingual news collection covering the 2021 Tokyo Olympics. The collection consists of articles written in different languages and labeled based on their reporting events. We then develop a new online clustering algorithm called Wasserstein-based news Article Clustering (WAC). This two-stage, distance-based algorithm uses Wasserstein distance to analyze the contextual and temporal similarities between clusters and decide when to merge them. The algorithm is tested on two data sets and compared with other online news clustering algorithms, showing that WAC performs comparably to the best-performing supervised algorithms without requiring any prior fine-tuning.

Finally, we define multi-modal data fusion of heterogeneous data sources. We focus on predicting stock market dynamics by combining text and data streams. We develop four methods for including text information in time series forecasting, which are then used to predict the stock close and daily volatility dynamics. We also test using different textual representations. The experiments show that including multi-dimensional text representation can improve predictions when the input data is appropriately processed and the right text inclusion strategy is used.

Povzetek

Doktorska disertacija raziskuje uporabo metod strojnega učenja za analiziranje tekstovnih podatkov v zaporedju. Ti podatki so urejeni v določenem zaporedju, kjer ima vrstni red pomembno vlogo. Primeri vključujejo (1) stavke, kjer sta raba in vrstni red besed določena s slovničnimi pravili, (2) poročanje novic o dogodkih, ki so se zgodili ob različnih časih, in (3) dokumente, ki poročajo o tečajih delnic na finančnem trgu. Analiza teh podatkov nam pomaga pri razumevanju vzorcev, napovedovanju in odločanju pri različnih nalogah.

Najprej se osredotočimo na merjenje podobnosti besedil, kjer upoštevamo njihovo strukturo. Predlagamo dve metodi, Language Model Earth Mover's Distance (LM-EMD) in Order-Preserving Wasserstein Score (OPWScore), ki upoštevata kontekstualni pomen besed in njihov položaj v besedilu. Model LM-EMD je namenjen medjezikovnemu iskanju informacij in meri ustreznost dokumenta glede na dano poizvedbeno besedilo. Metrika OPWScore pa je namenjena ocenjevanju uspešnosti modelov generiranja besedil, kot so strojni prevajalniki. Obe metodi uporabita jezikovne modele za ustvarjanje kontekstualnih vložitev besed. Te vložitve se nato uporabijo za merjenje podobnosti besedil z uporabo teorije optimalnega transporta, zlasti Wassersteinove razdalje. Metoda OPWScore pri tem tudi omeji izračun na besede, ki so na podobnih položajih, s čimer poudarja vlogo umestitve besed. Metode smo ovrednotili in primerjali z drugimi modeli in metrikami, ki upoštevajo strukturo besedil. Rezultati kažejo, da upoštevanje njihove strukture prispeva k uspešnosti, ki temelji na tekočnosti besedila, hkrati pa vpliva na uspešnost, povezano z ustreznostjo besedila.

Nato raziskujemo uporabo modelov nevronskih mrež za gručenje novic v realnem času, upoštevajoč njihov čas objave. Najprej predstavimo postopek za ustvarjanje novih podatkovnih množic za gručenje novic. Ta avtomatizira zbiranje in gručenje novic, zaradi česar je olajšano delo ročnemu označevalcu. Postopek uporabimo za ustvarjanje nove čezjezične množice novic, ki poročajo o olimpijskih igrah v Tokiu 2021. Množica je sestavljena iz člankov, napisanih v različnih jezikih in označenih glede na poročanje dogodkov. Razvili smo tudi nov algoritem za realno-časovno gručenje novic, imenovan Wasserstein-based news Article Clustering (WAC), ki temelji na rabi metrik. Algoritem analizira kontekstualne in časovne podobnosti med gručami novic z uporabo Wassersteinove razdalje in se nato odloči, katere naj združi. Algoritem je ovrednoten na dveh podatkovnih množicah in primerjan z drugimi algoritmi za gručenje novic. Rezultati kažejo, da algoritem WAC daje primerljive rezultate kot najboljši nadzorovani algoritmi, brez potrebe po predhodnem prilagajanju.

Nazadnje definiramo fuzijo večmodalnih in heterogenih virov podatkov. Posvetimo se napovedovanju borznih dinamik z uporabo tekstovnih in numeričnih tokov podatkov. Razvili smo štiri metode za vključevanje tekstovnih informacij, ki jih uporabimo v modelih za napovedovanja dinamik zneska ob zaprtju in dnevnih volatilnosti trga. Testirali smo tudi rabo različnih tekstovnih predstavitev. Poskusi so pokazali, da vključitev več-dimenzionalnih reprezentacij besedil lahko izboljša napovedi, ko so vhodni podatki primerno obdelani in je uporabljena prava strategija vključevanja besedila.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

API	...	Application Programming Interface
BOW	...	Bag-of-Words
CLIR	...	Cross-lingual Information Retrieval
CNN	...	Convolutional Neural Network
EMA	...	Exponential Moving Average
EMD	...	Earth Mover's Distance (also *1-Wasserstein Distance*)
IDF	...	Inverse Document Frequency
IPS	...	International Postgraduate School
IR	...	Information Retrieval
ISDI	...	IoT Streaming Data Integration
IoT	...	Internet-of-Things
JSI	...	Jožef Stefan Institute
JSON	...	JavaScript Object Notation
LLM	...	Large Language Model
LM	...	Language Model
LSTM	...	Long Short-Term Memory
MA	...	Moving Average
MT	...	Machine Translation
NLG	...	Natural Language Generation
NLP	...	Natural Language Processing
NN	...	Neural Network
OOV	...	Out-of-Vocabulary
OPW	...	Order-Preserving Wasserstein
OT	...	Optimal Transport
RNN	...	Recurrent Neural Network
ReLU	...	Rectified Linear Unit
SA	...	Sentiment Analysis
SE	...	Sentence Embedding
TF	...	Term Frequency
URL	...	Uniform Resource Locator
WMA	...	Weighted Moving Average

Symbols

$\mathbb{R}$	...	set of real numbers
$\mathbb{R}^n$	...	space of n-dimensional real vectors
$\mathbb{R}^{n \times m}$	...	space of $n \times m$-dimensional real matrices
$\mathbb{R}^{n \times m}_{\geq 0}$	...	space of non-negative $n \times m$-dimensional real matrices
$\lVert \cdot \rVert_2$	...	ℓ^2-norm, also known as Euclidean norm
$\langle \cdot, \cdot \rangle$	...	scalar product
δ_x	...	Dirac delta function concentrated at point x
e^x	...	exponential function
$\mathbb{O}^n_x$	...	data stream
$\mathbb{O}^{i,j}_x$	...	data stream window
$\mathbb{O}^n_{x,\text{aggr}}$	...	aggregated data stream
$\mathbb{O}^n_{x,\text{sampler}}$	...	resampled data stream
$\mathbb{O}^n_{x,\text{mapper}}$	...	mapped data stream

Chapter 1

Introduction

> It's the job that's never started that takes longest to finish.
>
> J.R.R. Tolkien

1.1 Motivation

Textual data is a resource that consists of information expressed in natural language. It can vary in structure and be in different languages, where grammar delegates the order in which the words must appear in the sentence. Because of this, text can also viewed as sequential data. Due to its diverse content, textual data has been used to develop models and tools for text-related tasks such as information extraction, clustering, understanding, and generation. Recent research has led to the creation of large language models that can solve various tasks with high precision and minimal effort. However, while these models are effective at analyzing individual resources, they may not be suitable for solving tasks requiring simultaneous analysis of large data sets or data where sequential order is important.

Sequential data refers to data collected in a specific order or sequence, where the order of observations carries important information. Each data point is linked to a timestamp or particular position in the sequence, indicating its occurrence time or order relative to other data points. While we commonly associate sequential data with time series data, where each data point has a corresponding timestamp, they can manifest in various formats and contexts, including (1) natural languages, where word usage and order must adhere to the rules of grammar, (2) news publishing, where articles may reference multiple events occurring at different timestamps, and (3) finances, where stock prices are tracked and analyzed over time.

Analyzing sequential data allows us to identify patterns in data, make predictions, and make informed decisions based on the order of observations. These capabilities are important across different fields and applications, as such analysis can help us to identify dependencies within the sequence, discern underlying patterns, and make predictions, among others. These discoveries can be leveraged to improve solutions for various tasks, including the following case studies explored in this dissertation: (1) text generation and translation, where the output text has to be factually accurate and grammatically sound, (2) event detection in news articles, grouping articles that describe the same event, and (3) stock market prediction, determining future market dynamics.

Although the case studies come from different domains, the tasks require methods

with shared functionalities. They need to handle sequences and account for the internal structure of the data, which can be done by combining models, such as Transformers and Long Short-Term Memory (LSTM), with distances suitable for comparing sequential data. Furthermore, the methods must handle various difficulties presented by the data sequences. For instance, to combine or compare two data sequences, one must first align them, which is not trivial when the sequences are of different lengths and modalities or are not aligned by time. Textual data is also conventionally represented by a single vector of values; with this representation, the word sequence in the text is lost. Preferably, a text should be described as a collection of ordered word vector representations, which are then used in the analysis. Solving these difficulties would enable a more advanced sequential data analysis, leading to more valuable insights.

1.2 Goals, Hypotheses and Approaches

This dissertation aims to develop machine learning methods for analyzing textual data that also consider the sequence of the data. The developed methods consider two types of sequences: (1) the inner-data sequence, such as word order in the text, and (2) the temporal sequence of the data points. Both sequence types are important for analyzing textual data: The inner-data sequence relates to grammar, while the temporal sequence is crucial for comparing text from real-time streams, such as news and sensor data. The main goals of the thesis are the following:

Goal 1 **Method for semantic text similarity considering word order.** Development of methods for measuring sentence similarity that include the word sequence information into consideration.

The goal is to refine how we measure semantic similarity between sentences by considering the meaning of individual words and their sequence. Most traditional similarity measures do not consider sentence structure, which can lead to overlooking and emphasizing words that are not necessarily relevant to the given task. Developing such methods could improve the performance of machine understanding and interpretation of texts, which is important for applications like text generation and machine translation.

Goal 2 **Method for news stream clustering.** Development of a novel news clustering approach that uses different similarity methods.

In this goal, we try to develop innovative methods for news stream clustering, focusing on how news articles are grouped based on their content. This could be useful for processing and managing large-scale news data and thus allow better information retrieval, easier navigation through news topics, and potentially help identify facts and compare how news publishers report on the same event.

Goal 3 **Method for combining text and data streams.** Development of methods for combining text and data streams to improve data stream predictions.

To achieve this goal, we aim to develop various methods that merge text streams, such as news, with data streams to improve forecasting models' predictive capabilities. This interdisciplinary approach could benefit domains such as financial stock forecasting, as text data could provide rich information complementary to that provided or found in data streams.

Goal 4 **Evaluation.** Evaluation of the proposed methods on various data sets.

Finally, this goal tests the developed methods across various data sets to ensure their effectiveness and robustness. Evaluation is important to confirm that new methods and approaches perform well in different scenarios and find areas where the methods can be improved.

In addition to the goals, we form three main hypotheses that will be experimentally tested in this thesis:

Hypothesis 1 Text similarity measures that include word sequences provide more contextually aware scores.

The hypothesis challenges traditional text similarity measures, which usually ignore sentence structures and might miss crucial semantic details. It proposes a direction for more contextually aware text analysis by considering sentence structure. This could enhance tasks that depend on a deep understanding of text content, such as text generation, machine translation, and summarization, where word order significantly affects meaning.

Hypothesis 2 Using similarity metrics that include structure in news stream clustering algorithms improves the quality and homogeneity of the generated news clusters.

This hypothesis highlights the importance of using advanced, structure-inclusive similarity metrics in news clustering algorithms. It challenges traditional clustering approaches that compare aggregated representations and suggests that using the entire structure and information of news improves the quality and homogeneity of the created clusters. Better clustering can enhance automated monitoring systems that depend on precise news aggregation to track and analyze global events.

Hypothesis 3 Including relevant textual information in numeric data streams improves the performance of data stream predictions.

The hypothesis highlights the integration of text and data streams to enhance the predictive capabilities of models dealing with data streams. This is particularly important in areas where real-time data is critical, such as finance trading. Text data can offer information often missing in data streams, potentially leading to more accurate forecasts and helping decision-makers respond more effectively to emerging trends and situations.

To achieve the goals outlined above and to test the validity of the hypotheses, we follow the methodology as described below:

Analysis of the current state-of-the-art in the field. We collected the scientific resources published in the last years that are relevant to our research topic. If we do not find sufficient resources, we extend the time interval in which it must be published. Afterwards, we read through the collection and wrote down the methodologies, parameters, data sets, and evaluation process used in the corresponding resource. These findings were then used in the subsequent steps of the methodology.

Acquisition of relevant data sources and preparation of a new data set for evaluating approaches. Given the information retrieved from the related work, we identify and retrieve the relevant data sources on which we would evaluate the proposed approaches. If the existing data sets do not meet the sufficient requirements for our evaluation, we design and create our own data set; such was the case for creating the novel temporal cross-lingual news article clustering data set.

Extensive analysis on aligning two data sequences to compare and combine data. We use the related work to analyze the approaches for aligning textual and temporal sequence data. Because of the lack of resources for analyzing textual data, we investigated approaches for comparing sequences of other data modalities, such as video and audio. For temporal sequences, we analyzed the temporal sequence data alignment approaches, mainly in the domain of time series. This analysis formed the basis on which we have designed and developed the novel approaches.

Implementation and evaluation of the proposed approaches. We developed novel methods and evaluated their performance on the acquired data sets. The evaluation approach is selected accordingly to reflect the model capabilities we wish to analyze correctly. We compare our method with the state-of-the-art methods and perform a comparative study. The evaluation and comparison results are displayed in tables and figures, highlighting the best performances.

Critical judgement of the method's performance, including the trade-off between the time complexity of the proposed approaches and overall performance. The evaluation results of the novel method are critically analyzed to determine the benefits and drawbacks of the methods. These are analyzed and presented in the discussions. Furthermore, we analyze the tradeoff between the inference time requirement and the overall performance. We suggest where and when to use the proposed methods based on the analysis.

1.3 Scientific Contributions

In this section, we describe the scientific contributions derived from the scientific research done within this thesis.

SC-1 *Novel unsupervised methods for measuring semantic text similarity focusing on the meaning and order of individual words.*

The novel method combines the semantic representations of words returned by Transformer-based language models with a modified optimal transport measure, which preserves the word sequence when calculating semantic text similarity. It is compared to other text generation and translation evaluation methods. Our method is more sensitive to fluency-based modifications than other unsupervised baseline methods while preserving adequacy-based performance. This relates to Hypothesis 1.

SC-2 *A new temporal data set consisting of labelled multi-lingual news articles intended for evaluating news stream clustering approaches.*

The new data set comprises news articles written in ten different languages on different topics. Each article contains its title, body text, URL, time of publication, language, news publisher, and cluster ID. Articles with the same cluster ID are presumed to describe the same event. The data set can be used to evaluate news stream clustering approaches related to Hypothesis 2.

SC-3 *A novel news stream clustering approach that uses the Transformer-based language models and an unconventional method for measuring text similarity.*

The novel clustering algorithm combines the semantic text representations from the language models with the Optimal Transport-based text similarity metrics to group news articles into multilingual clusters. It is compared to other online news clustering algorithms.

Our clustering algorithm performs similarly to the classifier-based clustering algorithms. This scientific contribution relates to Hypothesis 2.

SC-4 *An extension of the formal definition of data fusion of heterogeneous streaming data sources, which includes text data, and a method for using texts for data stream predictions.*

The extension combines one-dimensional and multi-dimensional features in data fusion. It generalizes the aggregate function, where an aggregate can be a predefined function or a function learned using machine learning approaches. The extension is then used to define models that learn abstract text representations to support the data stream forecasting in the financial domain. The addition of the text stream has been shown to improve the performance of the baseline forecasting model. The scientific contribution relates to Hypothesis 3.

SC-5 *Evaluation of the proposed methods on various data sources and real-world scenarios.*

Each method was evaluated and compared against state-of-the-art baselines, giving us insight into the developed methods' performance, advantages and disadvantages. These are discussed at the end of the method's corresponding chapter. This relates to all three Hypotheses.

1.4 Organization of the Thesis

The remainder of the thesis is structured as follows. First, the thesis presents the related work on non-sequential and sequential textual data analysis in Chapter 2, which covers measuring textual similarity, text stream clustering algorithms, and multi-modal stream forecasting. The chapter concludes with a critical summary of the related work.

It then describes optimal transport and how it is used for comparing sequential data in Chapter 3. It first provides an optimal transport overview in Section 3.1, covering its theoretical background. Next, Section 3.2 presents its extension, called order-preserving optimal transport, which can be used for measuring the similarity of sequential data.

The exploration and development of the novel text similarity methods are presented in Chapter 4. It first explores optimal transport for measuring cross-lingual semantic text similarity in Section 4.1. This inspired the exploration of the use of Optimal Transport for tasks where the sequence structure is important, such as text generation and translation tasks. This resulted in developing the method presented in Section 4.2. The chapter concludes by discussing the proposed similarity metrics in Section 4.3.

Next, the topic of online text clustering with neural networks is described in Chapter 5. The chapter first introduces an approach to creating multilingual news clustering data sets in Section 5.1, which was used to create a data set comprising articles reporting on the 2021 Tokyo Olympics. Next, a novel online news article clustering algorithm that uses Optimal Transport-based similarity measures to group articles is presented and evaluated in Section 5.2. The chapter concludes with a discussion of the proposed approaches in Section 5.3.

Chapter 6 focuses on streaming multi-modal fusion approaches. The chapter first presents a formal definition of data fusion in the Internet of Things domain in Section 6.1. Next, Section 6.2 presents an extension of the data fusion definition to include multi-dimensional features, such as text embeddings returned by language models. The extension is then applied and evaluated on stock price forecasting data sets. The findings in this chapter are then discussed in Section 6.3.

We conclude the thesis with Chapter 7, which presents the scientific contributions in Section 7.1, discusses whether the set goals are achieved and the hypotheses are confirmed in Section 7.2, and introduces further research work in Section 7.3.

Chapter 2

Related Work

> Every so often, you have to unlearn what you thought you already knew, and replace it by something more subtle.
>
> Terry Pratchett

This thesis builds upon prior research on the topics of non-sequential and sequential textual data analysis. While the non-sequential aspect disregards word or document order, it offers methods that can be expanded for sequential analysis of text. For example, while a method for comparing two news articles does not need to consider word order, it can still serve as a component for determining if the articles describe the same event, considering the publication datetime.

This section reviews the state-of-the-art methods and models across three domains. Section 2.1 explores methods for measuring textual similarity, emphasizing text representation, similarity in information retrieval, and evaluation metrics for natural language generation. Next, Section 2.2 examines algorithms for clustering text streams and available data sets for evaluating such algorithms. Lastly, Section 2.3 reviews methods for streaming data and text fusion and time series forecasting models.

2.1 State-of-the-Art on Measuring Textual Similarity

This section provides an overview of state-of-the-art methods used to measure textual similarity. Section 2.1.1 presents various methods for converting raw text into machine-readable objects, which are building blocks for natural language processing tasks. The methods for measuring text similarity in the context of (cross-lingual) information retrieval are presented in Section 2.1.2, while methods used to measure text similarity in the context of natural language generation are presented in Section 2.1.3.

2.1.1 Text representations

Before a text undergoes machine-learning processing, it must first be transformed into a mathematical object that computers can easily manipulate. In most cases, the objective is to extract valuable information from the text, which can be applied to a variety of tasks, including measuring text similarity.

Text representation is crucial in converting unstructured text into vector representations. These vectors can convey statistical information about the words' occurrences, lexi-

cal information (connected to nouns, pronouns, adjectives, etc.), or semantic information, capturing the meaning of (sub)words.

Over the years, many techniques have been developed, ranging from simple word counting to sophisticated neural networks, such as Large Language Models (LLMs). This section covers early and current approaches and models for generating text vector representations.

Early approaches

Early techniques relied on vector representations based on word statistics. One such approach was the Bag-of-Words (BOW) method, where a text document was represented as a collection of words, disregarding the order and structure of the text. Each word was treated as an individual feature, indicating binary values if the word was present or absent from the document. Alternatively, the feature could encode a combination of its term frequency TF and inverse document frequency IDF [1], [2]. The BOW could also be extended to include sequences of n words, known as n-grams. This extension allowed for the formulation of features naturally formed from multiple words (such as "New York" and "Golden Gate Bridge"), thus addressing more complex natural language processing tasks.

Early text representation techniques faced challenges. The "curse of dimensionality" emerged due to the large vocabulary size required to encode a diverse set of words, leading to vectors with more dimensions than samples in the data set. This increased vector sparsity, computational complexity, overfitting during model training, and higher data storage and memory demands. Moreover, synonym words were treated as distinct features in the BOW model, complicating the comparison of text documents with similar content but different wording.

To address these challenges, text documents were typically pre-processed before creating vector representations. Operations like lemmatization (reducing words to their base or dictionary form), stemming (reducing words to their root form by removing the prefixes and suffixes), and stopword removal (eliminating commonly used, but often irrelevant, words) were employed. While pre-processing mitigates the "curse of dimensionality," it does not fully resolve the synonym issue. Researchers sought a solution, which eventually materialized through word embedding models.

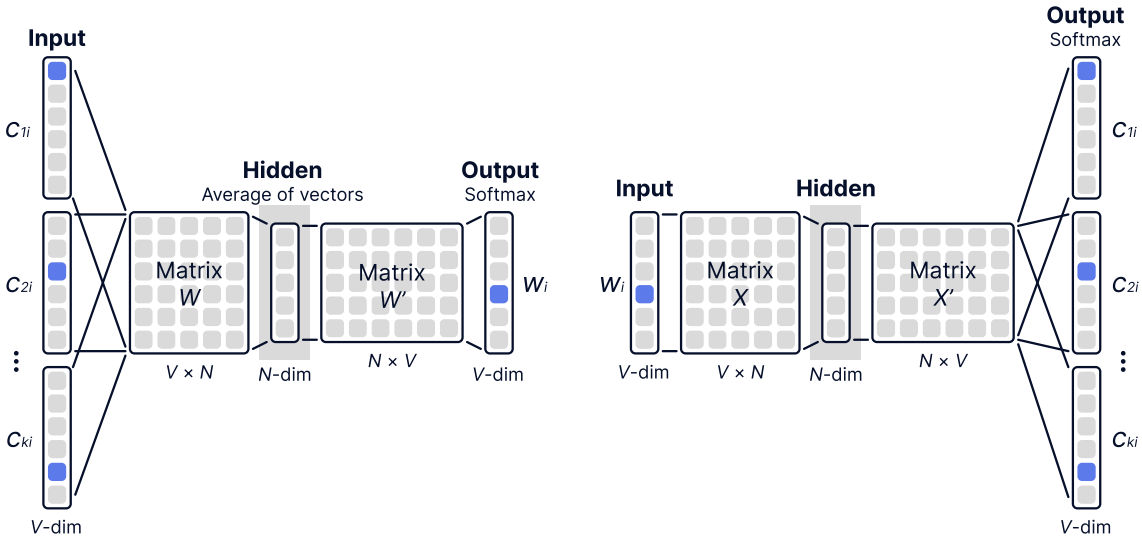
Word embeddings

The concept of modern word embeddings was introduced by developing the word2vec model [3]. This neural network model offers vector representations of words in a high-dimension space, capturing both semantic and syntactic relationships between words. Essentially, words with similar meaning and contextual usage tend to have similar vector representations.

Word2vec is comprised of two model variants: the Continuous Bag of Words (CBOW) and Skip-Gram. The CBOW variant predicts the target word based on its surrounding context words, while the Skip-Gram variant predicts the context words given the input target word. Both variants utilize a shallow neural network with a single hidden layer, where the hidden layer weights serve as the word embeddings. Figure 2.1 illustrates the architecture of both model variants.

Similar vector representations to word2vec are generated using GloVe [4], which employs matrix factorization techniques. The model utilizes a word co-occurrence matrix to construct a low-dimensional representation for each word, capturing both local context and global co-occurrence statistics.

However, a drawback of both word2vec and GloVe is their inability to handle out-of-vocabulary (OOV) words, i.e. words not present in the training corpus. Extensions like



(a) The CBOW model architecture. The hidden state is the average of the $W^T c_{ji}$ vectors, where c_{ij} are the words found in the context window. The output vector is compared with the one-hot encoding vector of the target word w_i.

(b) The Skip-Gram model architecture. The hidden state is equal to the $X^T w_i$ vector. The output vector is compared with all one-hot encoding vectors of the context words $c_{1i}, c_{2i}, \ldots, c_{ki}$.

Figure 2.1: The word2vec model architectures. Both model variants require a context window from which the target word w_i and the context words $c_{1i}, c_{2i}, \ldots, c_{ki}$ are taken. Values V and N determine the input, hidden, and output vector size.

FastText [5] have been proposed to address this limitation. FastText treats words as a bag of character n-grams (subword units) and predicts the probability distribution of subword units, instead of whole words, during training. This approach allows FastText to handle unseen and OOV words better by finding similarities based on shared subwords.

While these word embedding models are effective in a monolingual setting, they do not support the comparison of cross-lingual texts. To address this issue, various methods align word embeddings trained on different languages into a common semantic space [6]–[10]. This generally involves identifying anchor terms, i.e. words with similar meanings in different languages, and aligning them in the language embedding spaces using linear transformation. The anchor terms are identified either beforehand or based on the term statistics. Figure 2.2 illustrates the idea of aligning embedding spaces.

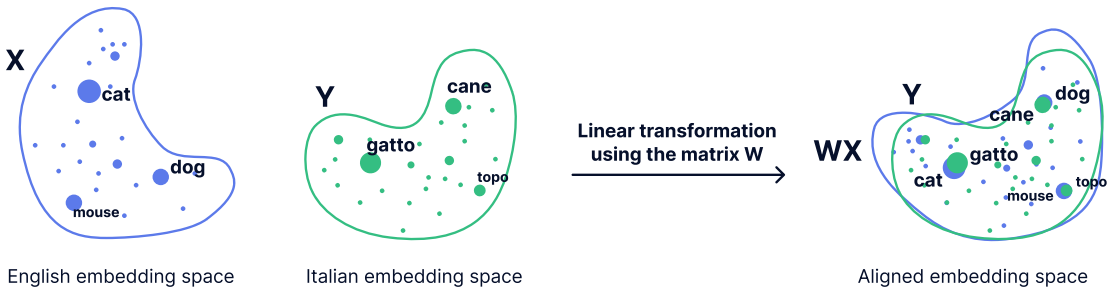


Figure 2.2: The illustration of aligning embedding spaces based on selected anchor points, adapted from previous work [8].

There are different ways one can create document vector representations using word embeddings. One way is by aggregating the embeddings of the words that appear in the document [11]. Another way is by using doc2vec [12], an extension of the word2vec, which produces a fixed-length vector representation of each document in a corpus. Both approaches produce vector representations that can be used for different document-related natural language processing tasks.

All word embedding approaches share a limitation: the vector representations remain static regardless of the context in which a word appears. For example, the vector representation of the word "mark" will be the same whether it means "grade", "target", "sign" or something else. Language models addressed this limitation, which is discussed in the following section.

Language models

Language models (LMs) are probabilistic models that assign probabilities to word sequences. This functionality allows LMs to predict the most likely word to follow in a given context, making them versatile for tasks such as text classification, text generation, and text similarity.

Historically, LMs were built using word co-occurrence statistics and recurrent neural networks [13]. However, recurrent neural networks face challenges in capturing long-range dependencies between words. This limitation was addressed by the Transformer model [14] depicted in Figure 2.3. Its key component is the attention mechanism, which enables the model to consider the entire input sequence when identifying the dependencies and relationships between the words.

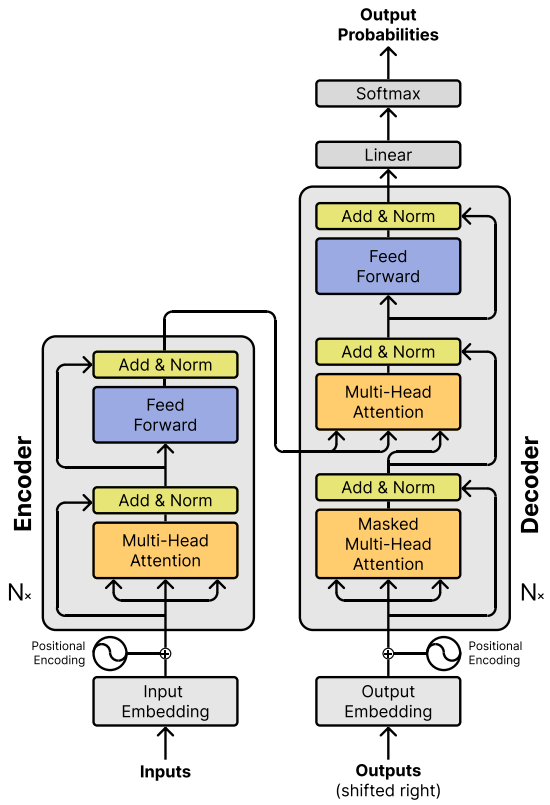


Figure 2.3: The Transformer model architecture, following the encoder-decoder structure, as presented in the original paper [14].

The Transformer's initial results inspired further exploration, leading to encoder-only and decoder-only models.

Encoder-only models were shown to perform well on text understanding tasks, such as question answering, named entity recognition, and semantic textual similarity. They encode the input text into a sequence of hidden representations and use these representations through a classification layer to solve the task at hand. Examples of encoder-only models are BERT [15], XLNet [16], and RoBERTa [17]. Since these models can have an increased number of parameters, methods for reducing the model size while retaining the model's performance have been developed [18], [19]. While most of the research focuses on monolingual tasks, there is increasing interest in the model's capabilities of processing multi-lingual and cross-lingual texts. As a result, cross-lingual encoder-only models, such as mBERT [20], XLM [21], and XLM-RoBERTa [22], were developed and applied for solving text understanding tasks involving texts in different languages.

Decoder-only models are used for text generation tasks, such as machine translation, text summarization, and recently also question answering. They first encode the input text into a sequence of hidden representations, which are then utilized to produce a sequence of output tokens, over which a probability distribution is calculated to determine the possible next words. Examples of such models are GPT [23], GPT-2 [24], and GPT-3 [25].

Moreover, extensive investigations were conducted into the capabilities of encoder-decoder models. Encoder-decoder models, such as BART [26] and T5 [27], were shown to be particularly good at tasks that require both understanding and generating text, such as machine translation, text summarization, and question answering.

Recently, language models have been significantly scaling in both parameter size and training data, resulting in the so-called Large Language Models (LLMs). One such model is Galactica [28], an LLM trained on scientific papers. Another study [29] utilized reinforcement learning through human feedback (RLHF) [30], [31] to fine-tune the GPT-3 model, resulting in enhanced text generation capabilities. This approach played a crucial role in the development of ChatGPT[1], a conversation agent released in November 2022. Since then, various general LLMs were developed, including Llama [32], Llama-2 [33], Falcon [34], Mistral [35], PaLM2 [36], and Gemini [36]. Additionally, specialized models have been introduced for finance [37], [38], coding [39], and medicine [40]–[43], among other.

Despite demonstrating strong performance across various tasks, experiments have been conducted to assess the intrinsic capabilities of these models [44]–[49]. Due to the resource-intensive nature of LLMs, researchers are exploring new approaches to enhance the model performance. This includes monitoring the model training process [50], modifying the model architecture [51]–[53], and refining the training procedures of LLMs [54]. More information can be found in a comprehensive survey on recent open-source LLMs [55].

2.1.2 Text similarity and information retrieval

The objective of an efficient Information Retrieval (IR) method is to retrieve documents relevant to the user's query. Text similarity methods play a pivotal role in information retrieval by determining which documents are most similar or relevant to the user's query. This section explores various approaches to measuring semantic text similarity in the context of information retrieval.

Distance metrics

One of the most common approaches to measuring the similarity between texts is using distance metrics. To employ these metrics, one must first create vector representations,

[1] `https://chat.openai.com/`

typically using one of the approaches outlined in Section 2.1.1. The generated vector representations exist in a continuous vector space. Afterward, a metric is applied to measure the distance between the two vectors, where texts with closer vector representations are considered more similar.

Distances, such as Euclidean and cosine distances, are commonly utilized where texts have single-vector representations. *Euclidean distance* measures the length of the line segment between two vectors $\vec{x}, \vec{y} \in \mathbb{R}^n$ in an Euclidean space, calculated as follows:

$$d_{\mathrm{E}}(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}, \quad \vec{x}, \vec{y} \in \mathbb{R}^n. \tag{2.1}$$

In comparison, the *cosine distance* measures the similarity between two non-zero vectors by computing the cosine of the angle between them. Because of this, the cosine distance depends only on the angle and not on their magnitude. The distance is computed as follows:

$$d_{\cos}(\vec{x}, \vec{y}) = 1 - \cos(\vec{x}, \vec{y}), \quad \cos(\vec{x}, \vec{y}) = \frac{\langle \vec{x}, \vec{y} \rangle}{\|\vec{x}\|_2 \|\vec{y}\|_2}, \tag{2.2}$$

where cos represents the cosine similarity. It is important to note that cosine distance is not a true distance metric as it does not exhibit the triangle inequality property[2].

It can be shown that the Euclidean and cosine distances are equivalent when the vectors $\vec{x}, \vec{y} \in \mathbb{R}^n$ are unit vectors, i.e., $\|\vec{x}\|_2 = \|\vec{y}\|_2 = 1$. Therefore, when the vector representations are unit vectors, either of the distances can be used to measure the textual similarity, yielding equivalent results.

Alternatively, a text can be represented as a set of vector representations, where the vectors correspond to individual words or tokens. While this representation is more complex, it allows more intricate text similarity measurements. One such approach involves using optimal transport [56], specifically Wasserstein distance, which measures the distance between two probability distributions. Although Wasserstein distance has been widely adopted in image processing and retrieval [57]–[59], it also finds applications in natural language processing, including model distillation [60] and automatic short answer grading [61]. In text similarity, Wasserstein distance can be utilized to measure the similarity of two sentences based on their associated word embeddings [62]. One method for calculating Wasserstein distance involves using the Sinkhorn-Knopp algorithm [63]. Section 3 gives an overview of optimal transport.

Query expansion

Query expansion is a technique which extends the query by adding values to improve document retrieval. The added values are usually similar or related to those in the query and can be assigned from knowledge bases [64], [65], ontologies [66], or by using word embeddings [67]. An extensive survey of query expansion techniques can be found in [68]. In addition to developing an effective model for finding relevant documents, research on other aspects, such as fairness of the retrieved documents [69], and the experiment reproducibility on neural information retrieval models [70], has been explored in the last years.

Cross-lingual information retrieval

To retrieve documents that are in various languages, one must bridge the lexical gap between them. Because of this gap, traditional information retrieval models need to be

[2] A metric $d: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ adheres to the triangle inequality if, for any three points $\vec{x}, \vec{y}, \vec{z} \in \mathbb{R}^n$, the following holds: $d(\vec{x}, \vec{z}) \leq d(\vec{x}, \vec{y}) + d(\vec{y}, \vec{z})$.

accompanied by methods that can transfer the documents and queries into the same or similar semantic space. The most commonly used are machine translation systems and word embeddings.

Machine Translation systems (MT) are used to translate either the query or documents [71]. They require huge amounts of parallel data to train the models where such resources are scarce for many language pairs. If needed, it is possible to employ MT services from different providers, such as Google Translate[3], DeepL[4], and Translate.com[5], but can be expensive when the data set is huge.

Word embeddings are used to generate query and document vector representations in the same semantic space. While most of them are trained on single language data sets, unsupervised approaches for aligning embeddings in a common vector space were explored [6], [8]–[10], [72]–[74]. The aligned word embeddings are then used to either (1) translate the query into the target languages and use traditional IR methods to find relevant documents, or (2) create a vector representation of both the query and document and calculate the cosine distance of the two vectors [75].

Language models for cross-lingual information retrieval

Language models are used to generate query and document vector representations. The majority of research is on modifying BERT [15] for monolingual information retrieval [76]–[82].

Let `[CLS]` be the sentence class token, representing the start of the text sequence, and `[SEP]` be the separator token, used to signal the end of the text sequence. Then, the models measure the document's relevance by concatenating the query and document text, i.e., constructing the input in the form of `[CLS] query [SEP] document [SEP]`, sending it through the model, and using the representation of the `[CLS]` token to predict the document's relevance. This approach enables the sharing of the query and document contexts with each other but is also expensive to compute since it has to be done for every query and document. For cross-lingual information retrieval, multilingual language models [20]–[22] have also been used to map texts in different languages in a common semantic space. For instance, multilingual BERT was used to generate inputs of the classification layer for predicting if a document is relevant to the query [83].

An alternative approach called SBERT [84] constructs the query and document representations separately by pooling or aggregating the language model's output. One can then create the document representations ahead of time and generate only the query vectors at run-time. The approach has been further extended by mapping the generated monolingual text embeddings of different language texts into a common semantic space via knowledge distillation [85] and thus making the embeddings multilingual. The extension was demonstrated for more than 50 languages from various language families and was shown to be efficient in cross-lingual information retrieval.

2.1.3 Text similarity as an NLG evaluation metric

The aim of a natural language generation (NLG) evaluation metric is to automatically assess the performance of text generation models used for text translation [86], [87], summarization [88]–[93], data-to-text generation [94], [95] and question answering [96], among others. This is done by measuring the similarity between the model's output and the gold reference prepared by human annotators. Furthermore, the metric-provided scores require

[3] `https://translate.google.com/`

[4] `https://www.deepl.com/en/translator`

[5] `https://www.translate.com/`

to be correlated to the scores given by human evaluators. To this end, various metrics were developed, most of which can be grouped into either using sequence matching or contextual embeddings.

Sequence matching metrics

The *n-gram matching metrics* measure the performance of the text generation model by matching sequences of n-consecutive words (called n-grams) between the generated output and its corresponding gold reference. The metrics differ based on how the matching is performed, which includes counting the number of n-grams between the two sentences [97], measuring the lengths of common sub-sequences [98], comparing the positions of 1-grams, their stemmed forms and semantic meaning [99], and by measuring the cosine distance between the n-gram-generated sentence representations [100].

One of the drawbacks of most of these metrics is that they do not consider the semantic meaning of the words, thus two sentences that use synonyms will have a lower score. This problem is being addressed by the metrics using contextual embeddings.

Embedding-based metrics

The *embedding-based metrics* use semantic word representations to measure the similarity between the generation model's output and its reference. Semantic representations enable assigning high similarity to words that have the same or similar meaning. Because of this, the embedding-based metrics can effectively assess the adequacy of the generated texts.

One of the more straightforward approaches to evaluate the generation model is BLEURT [101], a regression model where the input concatenates the model's output and its reference, and the output is the predicted score. More complex approaches acknowledge that the language model's layers provide different abstractions of the input text and use them to assess the quality of the generated text. One such metric is MoverScore [102], which utilizes optimal transport to generate the score based on the pairwise cosine similarities between the generated text and its reference. Another metric, also viewed as a stricter version of MoverScore, is BERTScore [103] which measures the pairwise cosine similarity between the tokens of the generated text and its reference, which are then used to compute the quality of the generated text. During the metric's ablation analysis, it was found that the metric's performance depends on the selection of both the language model and the specific hidden layer used to generate the semantic representations. COMET [104] measures the similarity by considering not only the generated text and its reference but also the input of the generation model. The metric generates a single embedding for each token by learning how to combine the representations of all language model's hidden layers, which is then used to assess the quality of the generated text. BARTScore [105] leverages the text generation capacities of the BART language model [26] to measure the quality of the text generation model's output. It enables different generation task evaluation by prefixing or postfixing to the source input text. However, the evaluation was performed only on English texts.

While the mentioned embedding-based metrics have been shown to have a high correlation with human judgement scores, they are all word-order agnostic, i.e. the word order in the sentences is not considered when computing the final score. However, the word order is important as it determines if a sentence is grammatically correct and well-structured, thus it is connected with the sentence's fluency and should be considered as part of the evaluation process [106].

2.2 State-of-the-Art for Text Stream Clustering

This section explores the current state-of-the-art data sets and methods utilized for text stream clustering. Section 2.2.1 offers an overview of the existing data sets used for training and evaluating text stream clustering approaches. Additionally, Section 2.2.2 presents the various algorithms designed for clustering text streams in both monolingual and multilingual environments.

2.2.1 Text stream clustering evaluation data sets

Manually annotated data sets are crucial in developing new text stream clustering algorithms and evaluating their performances. These data sets must capture the diverse volumes, varieties and velocities at which text streams can occur. Most publicly available data sets for text stream clustering predominantly comprise news articles. This section describes some of the most commonly used data sets for evaluating text stream clustering algorithms.

The TREC Microblog Track [107], [108] is a research initiative focusing on information retrieval and the analysis of microblog data. TREC data sets typically include tweets and associated metadata collections that vary across different years and editions of the TREC Microblog Track.

The AG News data set is a collection of over 1 million news articles gathered from over 2,000 news sources in more than 1 year of activity. The data set is available for research purposes, which includes text stream clustering [109]. The 20 Newsgroups data set [110] consists of 20,000 newsgroup documents, partitioned nearly evenly across 20 different newsgroups. This collection has become popular for text classification and clustering experiments. The Routers-21578 data set [111] comprises 10,369 Routers news articles with predefined categories, making it suitable for evaluating clustering algorithms. The TDT2 Multilanguage Text Corpus [112] includes news data collected daily from nine sources in two languages, English and Mandarin Chinese, for six months. It provides both manually-created reference text and automatically generated text for all broadcast and Mandarin data.

Event Registry [113] collects news articles from around 75,000 news sources and clusters them into multilingual event clusters. The event clusters and news articles are then available through their API service. A subset of this data set was then taken and prepared for training and evaluating news stream clustering algorithms [114]. This data set is now used as one of the main data sets for evaluating online news clustering algorithms. Alternatively, GDELT [115] contains over 200 million geolocated events based on news reports from various international news sources. Although it does not explicitly group news articles into event clusters, it provides a detailed record of each event, including date, location, involving actors, event type, and relevant tone information. A comparison between GDELT and the Event Registry was conducted [116], concluding that both data sets contain valuable information and are not interchangeable.

2.2.2 Online text stream clustering algorithms

Text stream clustering aims to group similar or related text data from a streaming context. Unlike traditional clustering, where the entire data set is accessible from the start, text stream clustering deals with continuous, dynamic, and potentially infinite text data streams. Various approaches can be employed for clustering text streams, depending on their variety and language setting. This section outlines the algorithms for clustering text streams depending on their structure.

Short text stream clustering

Social media provides a valuable environment for detecting real-world events. However, the text data published there is typically short and prone to the concept drift problem, where discussion topics quickly change and become outdated. In general, short text stream clustering can follow either the one-pass scheme, where texts are processed only once as they come one by one, or a batch scheme, where data arrives in batches and texts within the batch can be processed multiple times. This section presents the methods for clustering data from short text streams.

A recent survey on short text stream clustering provides a comprehensive overview [117]. For completeness, we provide a brief overview of the different approaches for short-text stream clustering. These algorithms typically fall into one of the following groups: The first group consists of *Dirichlet process-based methods*. These methods use a variation of the Dirichlet Process, a non-parametric stochastic process. They calculate the probability of generating a new cluster for an incoming text instance or assigning it to one of the existing clusters in their models. This can be achieved by leveraging co-occurrence of words [118] or single-word probability, where the process handles each word individually [119]–[121]. The second group consists of *similarity-based incremental methods*. These methods create text clusters based on various approaches for measuring similarities [122], including outlier detection and reassignment of outliers to clusters [123], as well as using a density clustering method to discover events with noise [124]. The last group consists of *word relation network-based methods*, which handle data streams in batches and construct a word relation network using the word frequency and word co-occurrence values [125], [126].

News stream clustering

A news stream comprising articles reporting on world events can have diverse structures, discuss single or multiple events, and be written in various languages. The goal of news stream clustering is to group articles that report on similar events. This section provides an overview of both monolingual and cross-lingual news clustering algorithms.

Monolingual clustering. When all of the articles in a stream share the same language, language-specific features can be used to assess how to group the texts. For instance, Event Registry [113] performs monolingual clustering using an online clustering algorithm based on [127], [128]. The algorithm uses various representations to capture the cluster characteristics, including vector representations of the article's title and body, as well as detected named entities, which include entities formed from Wikipedia concepts [129], [130]. In a related work [131], information extraction techniques, such as named entity recognition and part-of-speech tagging, along with temporal and content information, are used to construct the vector representation of the article. In both cases, cosine similarity is used to determine whether an article should be added to an event cluster.

Alternatively, newsLens [132] creates a graph of news articles connected based on their common keywords and uses a community detection algorithm to identify news events. The article's keywords are retrieved using TFIDF, where a pre-defined threshold is applied to retain significant keywords.

News articles often mention multiple events. In such cases, Transformer-based language models can be utilized for event sentence co-reference identification, which links parts of articles to multiple events [133]. It involves identifying if two sentences refer to the same event using a classification model utilizing BERT-based language models [15], [18]. Afterwards, it re-scores the sentence pairs considering the relation to other sentences and

their scores. Finally, the obtained scores of the sentence pairs are used to construct the event clusters.

Cross-lingual clustering. Clustering streams of text data in different languages is typically achieved through one of two approaches:

Two-step clustering. The two-step clustering approach initially forms monolingual article clusters by clustering articles within each language separately. Afterwards, these monolingual clusters are grouped into multilingual ones. Event Registry [113] utilizes trained Support Vector Machine (SVM) models [134], [135] for this purpose. Key features include cross-lingual cluster similarity, calculated using a statistical approach called Generalization of Canonical Correlation Analysis [136], as well as the time difference between clusters, time variability within clusters, and the similarity of the most frequently annotated entities in both clusters, represented using language-independent identifiers.

The emergence of models producing multilingual contextual embeddings has led researchers to explore their use in creating multilingual event clusters. In a study by Miranda et al. [114], monolingual article clusters are first created using language-specific features. Afterwards, multilingual word embeddings [7] are utilized to generate cross-lingual article representations, which are then employed to measure the similarity between monolingual clusters. Similarly, another related study [137] suggests extending the NewsLens algorithm [132] by incorporating the use of multilingual DistilBERT [19], [85] to merge monolingual into multilingual event clusters.

Single-pass clustering. The second, less common approach involves utilizing multilingual embeddings from the beginning and conducting cross-lingual event clustering in a single pass. In one such study [138], multilingual language models were employed to generate content representations of articles. Afterwards, a set of SVM-based ranking models was trained to determine whether an article should be assigned to a multilingual cluster based on its content and temporal representations.

2.3 State-of-the-Art for Multi-Modal Stream Forecasting

This section addresses state-of-the-art methods for integrating multi-modal data streams in the context of data stream forecasting. Section 2.3.1 presents an overview of approaches aimed at aligning multi-modal data streams, while Section 2.3.2 provides descriptions of algorithms employed in data stream forecasting.

2.3.1 Streaming multi-modal data fusion

The aim of streaming multi-modal fusion is to align multiple data streams temporally. Since data streams may produce values at varying frequencies, the initial step involves aligning them. This alignment can be achieved by incorporating domain knowledge into the models or employing domain-agnostic methodologies. This section outlines the various types of data streams and provides an overview of the general approaches used for their fusion.

Sensor data fusion

Sensor fusion helps improve certain functionalities and model accuracy in various domains, i.e. in positioning and navigation [139]–[141], in activity recognition [142], [143], in system monitoring and fault diagnosis [144]–[149], in transport [150] and others.

Most of the mentioned sensor fusion methodologies expect all the data to be coherent, available immediately and arriving in the correct order. In many localized systems, this is the case, while in others, such as the IoT scenario, data availability contributes to most of the problems. Rare contributions discuss handling delayed or out-of-sequence measurements [151]. Many of the systems also incorporate significant domain knowledge (model-driven approaches) into the data fusion model (mainly into the Kalman filter's transition matrix) [152], by which the models lose their generalization potential. Frameworks that have the potential to be applied in various use cases need to be domain knowledge agnostic (purely data-driven), at least with the modelling algorithm [153]. In this sense, any machine learning algorithm acts as a data fusion model since it combines multiple indicators into a single prediction. The idea has been developed further by heterogeneous feature fusion machines [139] that consider mapping multidimensional feature vectors into 1-dimensional output by using classic kernel functions, such as linear, polynomial and Gaussian with different regression methods. With these approaches, the challenge of generating correct and expressive feature vectors to support accurate modelling remains unsolved.

In large data sets, where stream mining is the chosen approach, data pre-processing and reduction are becoming the critical methodology for knowledge discovery [154], [155]. The authors identify the essential role of such systems for efficient machine learning models. Crucial identified pre-processing functionalities include concept drift detection and adaptation, missing data imputation, noise treatment, data reduction and efficient and accurate stream discretization algorithms. Automated data analysis is of no use if data pre-processing requires manual intervention [156]. The authors present adaptive pre-processing, which benefits the accuracy of the final prediction of real sensory data.

Text data fusion

Fusing text data with other modalities proves beneficial in addressing challenges across diverse domains. For example, in social security and stability, the identification of gambling websites involves independently and jointly processing image and text information on the website [157]. However, the integration of text and time series is predominately explored within the financial domain. In this context, text data, typically sourced from news articles, is analyzed, and specific information is extracted to enhance solutions for time series tasks. Researchers leverage text to detect latent causal structures and establish connections with time series events, explaining their occurrences [158].

Latent Dirichlet Allocation (LDA) [159] is a popular method used to extract topics from news articles, which are then utilized as features to improve the predictions of the financial time series [160], [161]. Additionally, to improve the forecasting of company stock volatility, company names were extracted from news events and processed by measuring the event-centric company similarities, which were added as a feature in the forecasting model [162]. Sentiment analysis is also employed to identify the sentiment of entities in the text, serving as features in predicting price movements [163]–[165].

An open-source library known as News signals [166] facilitates the conversion of textual data into time series signals. The library supports multi-document summarization, enables sampling of news data from entities, and facilitates the connection of entities with time series. While primarily designed for time series forecasting using textual data, the library is versatile and can be applied to various tasks related to data streams, including stream analysis, prediction, and causality.

2.3.2 Data stream forecasting algorithms

The data stream forecasting task is important in identifying potential future changes. It is essential to deploy algorithms and models capable of accurately detecting dependencies and relations within and between data streams to accomplish this. This section introduces various data stream forecasting models, encompassing incremental learning models, Transformer-based models, and graph neural networks.

Incremental learning models

Stonebraker et al. [167] have identified eight requirements of a stream processing engine (SPE) already in 2005. Among them are a requirement to handle stream imperfections (i.e. delayed or missing data), a requirement to integrate stored and streaming data, and requirements to keep the data moving, process the data, and respond instantaneously. Stream processing engines often base their modelling capabilities on incremental learning methodologies [168], [169]. The most popular method for incremental learning is still the Very Fast Decision Tree (VFDT) [170], which has been improved numerous times over the years. An interesting alternative, which can learn faster (achieve better accuracy sooner) and converge to batched decision tree form, is Extremely Fast Decision Tree (EFDT) [171]. Vertical Hoeffding Trees (VHT) [172] are the first distributed streaming algorithms for decision trees and offer significantly improved computation speed according to VFDT and EFDT. A lot of efforts are dedicated to incremental learning also in the deep learning domain [173]. With network architectures that include long short-term memory (LSTM) [174] modules, the problems of heterogeneous data fusion might be at least partially solved within the learning method.

Transformer-based forecasting models

Transformer-based models have gained significant traction in addressing time series tasks in recent years [175]. Specifically designed for forecasting, models such as Informer [176], Autoformer [177], and MQTransformer [178] specialize in both univariate and multivariate time series forecasting. However, their performance may lag behind simple linear models in certain time series scenarios [179]. This discrepancy is attributed to the use of the Transformer architecture, which, in natural language processing (NLP), typically employs a single token to represent (part of) a word containing rich contextual information. In contrast, for time series, a token comprises one or more values at a given timestamp, lacking the general contextual information inherent to time series data. Models like PatchTST [180] and iTransformer [181] have been developed to address this limitation. These models patch multiple timestamp values to create a token representation, leading to improved performance in forecasting tasks. Additionally, Transformer models have been extended to handle multi-modal inputs, specifically combining text and time series data for the data stream forecasting task [182].

In zero-shot time series forecasting, the foundation model TimeGPT [183] has been introduced. Trained on diverse data sets, TimeGPT can produce predictions across various domains, representing a significant stride in the time series domain analogous to efforts in NLP. Moreover, Large Language Models (LLMs) originally used for NLP have demonstrated surprising proficiency in zero-shot extrapolation of time series [184]. This suggests that LLMs trained on text exhibit some capacity to forecast time series values, although the full extent of this capability remains to be explored. Anticipating further exploration and development in this area is a reasonable expectation.

Graph neural network forecasting models

Graph neural networks (GNNs) have recently demonstrated high capability in handling relational dependencies, positioning them as excellent candidate models for time series forecasting. One contribution in this regard [185] proposes a general GNN framework tailored specifically for multivariate time series. This framework automatically extracts relations among variables, captures spatial and temporal dependencies within the time series, and employs graph learning approaches to learn and provide accurate forecasts jointly. Beyond forecasting, GNNs have exhibited impressive performance in various other time series tasks, including classification, imputation and anomaly detection [186].

2.4 Critical Summary of Related Work

This section provides a critical summary of the related work covered, highlighting both positive and negative aspects and suggesting potential improvements that serve as the foundation for the research presented in this thesis.

In text representation, Transformer-based language models have demonstrated efficacy in generating contextual embeddings at both token and sentence levels. Leveraging their performance, our experiments incorporate various Transformer-based models, namely for employing encoder-only models to generate semantic vector representations of the text. These representations are a fundamental component in the novel approaches outlined in the thesis.

Text similarity methods predominantly fall into two categories: distance-based, commonly utilizing cosine distance, or models specifically trained to measure similarity based on text-related embeddings. Our work explores the less common Wasserstein distance, focusing on (1) measuring similarity between texts in different languages, (2) incorporating word order in measuring text similarity, and (3) assessing similarities between groups of documents.

Publicly accessible datasets for evaluating text stream clustering algorithms are limited. Moreover, these datasets are often classified into a small number of clusters with coarse topics. Although we have identified a suitable data set for evaluating news stream clustering algorithms derived from news articles obtained through the Event Registry API [114], it is notable that this data set encompasses a small number of daily events. This limitation hinders its reflection of real-world scenarios. Therefore, in this thesis, we constructed a new multilingual news stream data set with article distributions more closely aligned with those encountered in the real world.

Furthermore, reported news stream clustering algorithms commonly rely on classifiers to determine if two articles report the same event. In contrast, our approach involves developing a news stream clustering algorithm that groups articles based on the similarity of their content embeddings utilizing unconventional similarity methods and integrating information extracted using already-available extraction models.

While sensor data fusion has been extensively researched, the fusion of text data and streams with time series remains relatively unexplored. The predominant approach involves extracting discrete information from text as a feature in time series tasks. Our research explores how we can effectively leverage most text information and incorporate it as a component in time series forecasting.

Chapter 3

Optimal Transport for Comparing Sequential Data

Sequential data, including textual data, can be compared using various approaches and measures. One such measure is the Earth Mover's Distance (EMD), a special case of the Wasserstein distance derived from the optimal transport theory. We use this distance in the dissertation to compare sequential data to the problems described in the subsequent chapters. This section presents the optimal transport theory and the associated measures. Section 3.1 provides an overview of the optimal transport theory. Furthermore, Section 3.2 introduces order-preserving optimal transport, an extension that tries to preserve the sequence of data points.

3.1 Optimal Transport Overview

Optimal transport (OT) is a powerful approach to comparing probability distributions [56], [187]. Suppose we have two Polish spaces[1] X and Y. We then define the *probability measure* as a real-valued function that satisfies the measure properties and assigns a numerical value between 0 and 1 to events or sets within a probability space. Given the probability measures α on X and β on Y, we define the *coupling of* (α, β) as a probability measure $\pi \in X \times Y$ such that $(\mathrm{pr}_X)_{\#}\pi = \alpha$ and $(\mathrm{pr}_Y)_{\#}\pi = \beta$, where $(\mathrm{pr}_X)_{\#}$ and $(\mathrm{pr}_Y)_{\#}$ are push-forward operators that project the input probability measure π onto the probability measure defined on the spaces X and Y, respectively. The *optimal transport problem* asks for the coupling that solves the minimization problem

$$\mathcal{L}_c(\alpha, \beta) := \inf_{\pi \in \Pi(\alpha,\beta)} \int_{X \times Y} c(x, y) d\pi(x, y), \tag{3.1}$$

where $\Pi(\alpha, \beta) := \{\pi \in X \times Y : (\mathrm{pr}_X)_{\#}\pi = \alpha, (\mathrm{pr}_Y)_{\#}\pi = \beta\}$ is the set of all admissible couplings, and $c : X \times Y \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ is a cost function that is at least semi-continuous, i.e., the function is discontinuous at a finite number of points. Note that when X, Y are discrete spaces, every cost function c, as defined above, is continuous.

When the probability measures are defined on the same space, the optimal transport problem induces a notion of a distance, or similarity, on the space of probability measures. Suppose X is a Polish space, and α, β are probability measures on X. Given the ground metric $d : X \times X \to \mathbb{R}_{\geq 0}$ and the cost function $c(x, y) = d(x, y)^p$, the *p-Wasserstein distance* is defined as

$$W_p(\alpha, \beta) := \mathcal{L}_{d^p}(\alpha, \beta)^{1/p}. \tag{3.2}$$

[1] A Polish space is a separable, completely metrizable topological space.

It can be shown that W_p truly defines a metric on the space $\mathcal{P}(X)$ of all probability measures on X [56]. In addition, optimal coupling π that solves $W_p(\alpha, \beta)$ gives us insight into how to redistribute the mass between α and β, which is helpful for understanding the practical meaning of the solution. Thus, the p-Wasserstein distance is a powerful tool for measuring distances between probability measures.

From the *Kantorovich duality* [188] it follows that the optimal transport problem admits a dual problem which is more tractable in terms of computational complexity. Specifically, in the case of discrete measures, the dual problem can be solved with linear programming. Let us state one possible version of the Kantorovich duality.

Let (X, α) and (Y, β) be probability spaces, and $c: X \times Y \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ be a continuous cost function, such that

$$c(x, y) \geq \phi(x) + \psi(y), \quad \forall (x, y) \in X \times Y, \tag{3.3}$$

for some continuous functions $\phi: X \to \mathbb{R}$ and $\psi: Y \to \mathbb{R}$. Let us define the set of all pairs of functions (ϕ, ψ) satisfying Equation (3.3) as

$$\mathcal{A}_{X \times Y}(c) = \{(\phi, \psi): c(x, y) \geq \phi(x) + \psi(y),\ \forall (x, y) \in X \times Y\}.$$

Then, the Kantorovich duality is shown as

$$\mathcal{L}_c(\alpha, \beta) = \sup_{(\phi, \psi) \in \mathcal{A}_{X \times Y}(c)} \left(\int_Y \psi(y) d\beta(y) - \int_X \phi(x) d\alpha(x) \right). \tag{3.4}$$

Thus, the solution of the dual problem induces a solution to the original optimal transport problem.

In practice, solving the optimal transport problem is often computationally intensive. To simplify it, we often use entropic regularization, which adds an entropy-based penalty, such as Kullback-Leibler divergence, to the original OT objective function. This approach makes the problem more computationally efficient and numerically stable but provides an approximate solution to the original problem. In the discrete and discretized continuous case, the regularized OT can be solved using the Sinkhorn-Knopp algorithm [189], which is an iterative procedure that alternates between scaling rows and columns of the transport plan matrix, as described in Algorithm 3.1.

Algorithm 3.1: The Sinkhorn-Knopp algorithm: `SinkhornKnopp`

Function: `Wasserstein`
Input : The distribution of the two inputs $a \in \mathbb{R}^n, b \in \mathbb{R}^m$; the cost matrix $D \in \mathbb{R}^{n \times m}$; the regularization factor $\gamma \in \mathbb{R}_{\geq 0}$; the number of iterations N
Output: The Wasserstein distance

$\mathsf{K} \leftarrow e^{-\frac{D}{\gamma}}$;
$\mathsf{u}^{(0)} \leftarrow \mathtt{ones(dim} = n\mathtt{)}$; `// Vector of size` n `filled with ones`
$\mathsf{v}^{(0)} \leftarrow \mathtt{ones(dim} = m\mathtt{)}$; `// Vector of size` m `filled with ones`
for $i \leftarrow 1$ **to** N **do**
  $\mathsf{u}^{(i)} \leftarrow a \oslash \mathsf{K}\mathsf{v}^{(i-1)}$; `// Element-wise division`
  $\mathsf{v}^{(i)} \leftarrow b \oslash \mathsf{K}^T\mathsf{u}^{(i)}$; `// Element-wise division`
end
$\mathrm{P} \leftarrow \mathtt{diag(}\mathsf{u}^{(N)}\mathtt{)}\mathsf{K}\ \mathtt{diag(}\mathsf{v}^{(N)}\mathtt{)}$;
return $\mathtt{trace}(D^T P)$;

3.1.1 Probabilistic view of optimal transport

The optimal transport problem can be defined using probabilistic terms. Given two probability spaces (X, α) and (Y, β), we define a *coupling of* (α, β) as a construction of two random variables (A, B) such that $\text{law}(A) = \alpha$ and $\text{law}(B) = \beta$, where for an arbitrary random variable X we can think of law(X) as the probability distribution of X. Note that a coupling of (α, β) always exists [56]. The probabilistic equivalent to Equation (3.1) is solving

$$\mathcal{L}_c(\alpha, \beta) = \inf_{(A,B)\in\Pi_p(\alpha,\beta)} \mathbb{E}\left(c(A, B)\right),$$

where $\Pi_p(\alpha, \beta) = \{(A, B) : \text{law}(A) = \alpha, \text{law}(B) = \beta\}$ is the set of all admissible couplings defined in probabilistic terms, and $c : X \times Y \to \mathbb{R}$ is the cost function [56]. Similarly, the p-Wasserstein distance has an equivalent probabilistic formulation written as

$$W_p(\alpha, \beta) := \inf_{(A,B)\in\Pi_p(\alpha,\beta)} \mathbb{E}\big(c(A, B)^p\big)^{1/p}. \tag{3.5}$$

This definition of p-Wasserstein distance defines a metric on the space $\mathcal{P}(X)$ of probability measures on X, which is shown in [56]). In order to use p-Wasserstein distance for measuring relevancy between random variables, we must first introduce the *Prokhorov distance*. One possible definition is

$$d_p(\alpha, \beta) = \inf\left\{\epsilon > 0 : \inf_{(A,B)\in\Pi_p(\alpha,\beta)} \mathbb{P}[d(A, B) > \epsilon] \le \epsilon\right\}.$$

Distance $d_p(\alpha, \beta)$ is precisely the minimum distance "in probability" between random variables distributed according to probability measures α and β [190]. For all $p \ge 1$, the Wasserstein and Prokhorov distances may be related by

$$(d_p)^2 \le W_p \le (1 + \text{diam}(X))\, d_p, \tag{3.6}$$

where $\text{diam}(X) = \sup\{d(x, y) : x, y \in X\}$ [191]. This relation gives the means for interpreting the relevancy of two random variables: Suppose we are given a metric space X, such that the diameter $\text{diam}(X)$ is sufficiently small, and the space $\mathcal{P}(X)$ of probability measures on X. Let α, β be probability measures on X, and A, B be random variables such that $\text{law}(A) = \alpha$ and $\text{law}(B) = \beta$. Then a smaller p-Wasserstein distance between α and β implies that the probability that B is lying in the complement of the ϵ-ball centered at A, $\mathbb{B}(A, \epsilon) := \{C : d(A, C) < \epsilon\}$, is sufficiently small. Thus, we can conclude that there must be some relation between the random variables A and B.

3.2 Order-Preserving Optimal Transport

The original OT formulation has been shown to be useful for measuring sentence similarity both for evaluating text generation models [102] and in information retrieval [192]. However, it is invariant to the sequence in which the data points are. In this section, we describe the order-preserving optimal transport [193] and how it can be used for measuring sequential data points.

For simplicity, we will work in the space of discrete probability measures. Let us define a set of sample points $S := \{x_1, x_2, ..., x_n \,|\, x_i \in X\}$, where X is a Polish space (e.g the space of of n-dimensional vectors $\mathbb{R}^n$). We can then associate a discrete probability measure to the set of points S as follows:

$$\alpha := \sum_{i=1}^{n} a_i \delta_{x_i},$$

where $a_i \geq 0$, $\sum_{i=1}^{n} a_i = 1$ and δ_{x_i} are the Dirac delta functions defined as

$$\delta_{x_i}(y) = \begin{cases} 1, & x_i = y, \\ 0, & x_i \neq y, \end{cases} \tag{3.7}$$

where $x_i, y \in X$ are sample points. Value a_i represents the weight corresponding to sample point x_i in discrete probability measure α. For example, weights $a_i = \frac{1}{n}$ correspond to sample points being uniformly distributed.

Since the above set S is finite we may consider an order $\leq$ on it. Let us denote this set as $(S, \leq) = [x_1, x_2, ..., x_n]$. We will name this pair $(S, \leq)$ as the sequence of "sample" points in X and number n as the length of the sequence. Given two sequences $S = [x_1, x_2, ..., x_n]$ and $S' = [y_1, y_2, ..., y_m]$, we can assign to them the probability measures

$$\alpha = \sum_{i=1}^{n} a_i \delta_{x_i}, \quad \beta = \sum_{j=1}^{m} b_j \delta_{y_j}. \tag{3.8}$$

We now take a specific order relation $\leq_{temp}$ on S, which we call the temporal order on sample points. We describe it in the following way: Let index i of sample point x_i represent the temporal position of its sample point. We assign to it value $\frac{i}{|S|}$, which we name the relative temporal position, such that

$$x_i \leq_{temp} x_j \text{ if and only if } \frac{i}{|S|} \leq \frac{j}{|S|}.$$

Let P be the matrix solution of the original discrete OT problem with respect to measures α and β defined in Equation (3.8). If we wanted the optimal coupling to also preserve the temporal positions of the sample points, matrix P should have some (local) homogeneity. We can measure the (local) homogeneity of P using the *inverse difference moment* defined as

$$I(P) := \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{P_{i,j}}{\left(\frac{i}{n} - \frac{j}{m}\right)^2 + 1}, \tag{3.9}$$

where $P_{i,j}$ are the values of matrix P, and n and m are the lengths of sequences S and S', respectively. Value $I(P)$ should be as large as possible to capture the temporal positions of sampled points in matrix P.

Since we want to preserve the temporal order in sequences, the ideal distribution of values in P would have large values along the diagonal and gradually decreasing values outside the diagonal; having larger values on the diagonal corresponds to the matching of sample points at similar temporal positions. We thus set the following matrix G as the prior distribution of the values of P:

$$(G)_{i,j} = g_{i,j} := \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{l(i,j)}{2\sigma^2}}, \quad l(i,j) = \frac{\left|\frac{i}{n} - \frac{j}{m}\right|}{\sqrt{\frac{1}{n^2} + \frac{1}{m^2}}} \tag{3.10}$$

Choosing the Kullback-Leibler divergence of P with respect to G, denoted as $KL(P\|G)$, as the entropy measure, we can state the (approximate) order-preserving optimal transport problem as

$$\mathcal{L}_c^{\epsilon, ord}(\alpha, \beta) := \min_{P \in \Pi(\alpha,\beta)} \Big(\langle P, D \rangle - \lambda_1 I(P) + \lambda_2 KL(P\|G) \Big). \tag{3.11}$$

The solution to the above problem, therefore, is the optimal approach of matching the two discrete probability measures α and β while preserving their temporal order. More details can be found in the paper that introduces sequence-preserving optimal transport [193].

Chapter 4

Semantic Text Similarity Using Sentence Structure

This chapter focuses on developing semantic text similarity methods, which deal with the meaning of individual words as captured by their context and their word order within the text. Section 4.1 presents a novel approach to measuring multilingual textual similarity in information retrieval based on token similarity and returns interpretable results. Next, Section 4.2 presents a novel metric for measuring the semantic text similarity, which considers both the word's meaning and the inner-word sequence within the sentence. Finally, Section 4.3 discusses the experiment results and the advantages and disadvantages of the research presented in this chapter.

Section 4.1 is based on the published work [192], while Section 4.2 expands on the findings from [194]. This chapter introduces two scientific contributions: [SC-1] novel unsupervised methods for measuring semantic text similarity focusing on the meaning and order of individual words, and [SC-5] evaluation of the proposed methods on various data sources and real-world scenarios.

4.1 Measuring Cross-Lingual Sentence Similarity with EMD

Due to global connectivity, we are communicating and collaborating with people from all around the globe. Because of that, we are required to prepare documents and resources in the language the target audience will be able to understand. For example, resources may be prepared in a specific language or a more widely spoken language, depending on the audience. Thus, the topic is presented in different languages. This especially applies to education and law. In education, a topic is described, adapted, translated, and published in various languages, which can be seen in the massive amounts of available courses and textbooks. In law, legislation and treaties are translated into the languages of countries the documents are relevant to, making them accessible to their lawyers and decision-makers. Because of the huge amount of multilingual documents, one requires an effective search solution.

Modern cross-lingual document retrieval models provide limited explanations about the document's relevancy to the query. With the introduction of transformer-based multilingual language models, cross-lingual information retrieval (CLIR) has become more interesting in research communities. More specifically, cross-lingual document retrieval focuses on finding documents relevant to the query, where the document and query can be in different languages. Because of the language difference, the models must resolve to more advanced approaches than simple keyword matching to measure the document's relevance. These usually involve extracting textual features and representations that are

language agnostic. The features are then used to output a single value, which signals how relevant a document is to a query. However, this value does not explain why the document is relevant.

Understanding why a document is relevant is helpful for both the development of document retrieval models and their usage. When developing the model, one would normally test its performance. For example, when the model does not return appropriate results, knowing why a document is assigned the given relevancy value is useful for understanding why the model does not work, thus speeding the development process. Once the model is ready, it can be made available for its users. Sometimes, the users do not necessarily know how to structure the query to find documents they are interested in. Seeing why a document is relevant gives insights into how to change the query to get the appropriate results. Thus reducing the time they spend searching.

This work aims to develop a cross-lingual document retrieval model that can calculate a document's relevance to the query with high precision and provide insight into why a document is relevant.

The main research objectives are to (1) evaluate the model's performance on data sets consisting of queries and documents in different languages (language pair data sets), where the languages have various degrees of similarity, (2) assess the degree of document score interpretability the model provides, and (3) analyze the impact the objective loss function used during training has on the performance of cross-lingual document retrieval models.

We propose a learning-to-rank model named *Language Model Earth Mover's Distance* (LM-EMD), which uses multilingual BERT to generate token vector representations and Earth Mover's Distance (EMD) to measure the document's relevancy to the query and provide an interpretable view into why a document is relevant.

The model is evaluated on five language pairs, three high-resource and two low-resource, using the Precision at 1 (P@1) and Mean Average Precision (MAP) metrics. The results show that the proposed model performs similarly to the selected comparing models on high-resource languages, but its performance decreases on low-resource languages. We also analyze how the selection of the objective loss function used during training affects the performance of the document retrieval models. We compare two commonly used loss functions, cross-entropy and pairwise ranking, and find that training the models using pairwise ranking yields better performance.

The remainder of the section is structured as follows. Section 4.1.1 describes the theoretical approach to measuring document relevancy using optimal transport. The proposed model is presented in Section 4.1.2, followed by its implementation details in Section 4.1.3. The setting in which the model is evaluated is described in Section 4.1.4. Finally, we present the experiment results in Section 4.1.5.

4.1.1 Measuring document relevancy using optimal transport

This section presents the theory used to measure the document's relevance to the query. It is calculated using optimal transport presented in Section 3.

First, we define the document and query as a collection of subword segments referred to as tokens. We then construct a fully connected graph $G = (V, E)$, where the vertices $V = \{t_i\}_{i=1}^{N}$ are the tokens in the document and query, and the edges $E = \{(t_i, t_j, w_{i,j})\}_{i,j=1}^{N,N}$ are weighted with the distance between the edge's endpoints $w_{i,j} = d(t_i, t_j)$. When the tokens are represented by vectors, the distance between them can be measured with the cosine distance defined in Equation (2.2). Because of its definition, the cosine distance is bound between 0 and 2. Hence, the diameter of graph G has the following upper bound:

$$\operatorname{diam}(G) = \sup_{i,j}\{d_{\cos}(\vec{v}_{t_i}, \vec{v}_{t_j})\} \leq 2,$$

where $\vec{v}_{t_i}$ and $\vec{v}_{t_j}$ are the vector representations of token t_i and t_j, respectively.

Let query q and document d be subgraphs of G, where their corresponding tokens $q_i \in q$ and $d_j \in d$ are vertices that form a fully connected subgraph. Then, both the query and document can each be represented with a probability measure on G defined as

$$\mu_q = \sum_{i=1}^{n} w_{q_i} \delta_{q_i}, \quad \nu_d = \sum_{j=1}^{m} w_{d_j} \delta_{d_j},$$

where $\delta_{q_i}, \delta_{d_j}$ are the Dirac delta functions as defined in Equation (3.7), w_{q_i}, w_{d_j} are the weights associated with the tokens q_i, d_j, and n, m are the number of tokens in q, d, respectively. Since the diameter $\text{diam}(G)$ is small enough, and we can always find a coupling (Q, D) of (μ_q, ν_d), we may use the criterion given by Equation (3.6). Thus, if the p-Wasserstein distance $W_p(\mu_q, \nu_d)$ is (sufficiently) small, probability $\mathbb{P}(D \in \mathbb{B}(Q, \epsilon))$ is high; meaning random variable D, associated with document d, has a high probability of lying in the ϵ-ball centered at the random variable Q, associated with query q. Hence, we can conclude that a small $W_p(\mu_q, \nu_d)$ value implies that document d is relevant to query q. In addition, because μ_q and ν_d are discrete measures, we can calculate the p-Wasserstein distance by using the optimal transport's dual problem formulation, which can be solved with linear programming.

4.1.2 The LM-EMD model architecture

Using the theory presented in Section 4.1.1, we propose a learning-to-rank model that generates interpretable document relevance scores. It consists of two components: (1) a multilingual language model used to extract the query and document token representations and (2) the EMD to calculate the document relevance scores. The EMD solution can also be used to interpret why the document is relevant. The model's architecture and data flow is depicted in Figure 4.1. The implementation details are presented in Section 4.1.3.

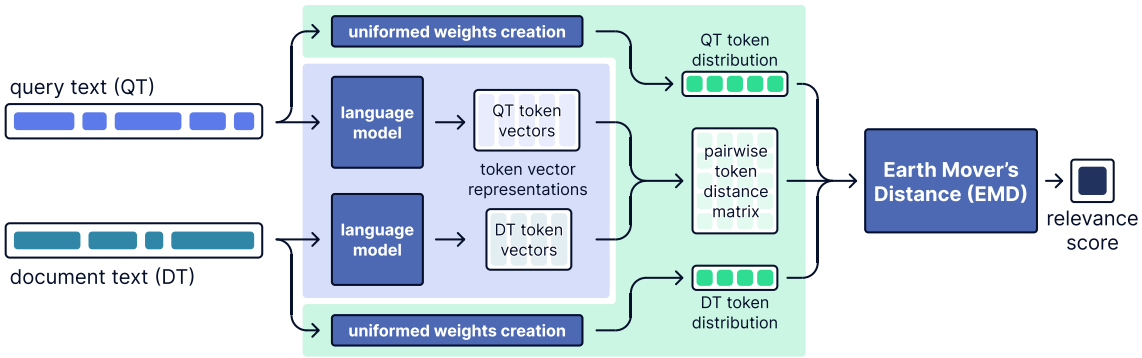


Figure 4.1: The LM-EMD model architecture. The first step consists of retrieving token representations used in the second step to calculate the Earth Mover's Distance between the query and the document.

Query and document representation

Language models generate contextual embeddings, i.e. representations that change based on the token's context. The language model first splits the input text into subword segments (tokens) using the model's associated tokenizer. This subword segmentation algorithm converts the input text into character n-grams. Afterward, the model generates a vector representation (an embedding) for each token based on the token's context. These

embeddings capture the token's semantic meaning. Therefore, tokens with similar or related meanings are found in similar contexts, hence their corresponding embeddings are closer in the embedding space. This can be extended to other languages by using multilingual language models, such as multilingual BERT [20], XLM [21], and XLM-RoBERTa [22].

Let q and d be the query and document, respectively. Using a language model, we extract the query's and document's tokens and their embeddings. Because the model has a length limit for its input, we split longer documents into parts such that each part has at most the maximum length the model can process, and afterwards, we concatenate the model's outputs.

The tokens and their embeddings are then grouped to create the query and document representation sets T_q, T_d consisting of (token, embedding) pairs used for measuring the document's relevance score:

$$T_q = \left\{(q_i, \vec{v}_{q_i})\right\}_{i=1}^{n}, \quad T_d = \left\{(d_j, \vec{v}_{d_j})\right\}_{j=1}^{m},$$

where n, m are the number of extracted tokens, q_i, d_j are the tokens extracted from query q and document d, and $\vec{v}_{q_i}, \vec{v}_{d_j}$ are the contextual embeddings of their corresponding tokens, respectively. Although the query and document can share the same tokens, their associated embeddings can differ since they reflect the token's meaning.

Measuring the relevance score

The 1-Wasserstein distance, also known as EMD, is used to measure the distance between two probability measures. We define the document relevance score as the EMD between the query and the document, such that a smaller EMD implies greater relevance. Let $D \in \mathbb{R}^{n \times m}$ be the matrix of distances between the query and document tokens. Similar to previous work [62], we use token embeddings and calculate their cosine distances:

$$D_{i,j} = d_{\cos}(q_i, d_j) = 1 - \frac{\langle \vec{v}_{q_i}, \vec{v}_{d_j} \rangle}{\|\vec{v}_{q_i}\|_2 \|\vec{v}_{d_j}\|_2} \geq 0. \tag{4.1}$$

Let μ_q and ν_d be the query's and document's respective token distributions:

$$\begin{aligned} \mu_q &= \sum_{i=1}^{n} w_{q_i} \delta_{q_i}, & \sum_{i=1}^{n} w_{q_i} &= 1, \\ \nu_d &= \sum_{j=1}^{m} w_{d_j} \delta_{d_j}, & \sum_{j=1}^{m} w_{d_j} &= 1, \end{aligned} \tag{4.2}$$

where δ_{q_i}, δ_{d_j} are the Dirac delta functions as defined in Equation (3.7), and the weights w_{q_i}, w_{d_j} represent the mass of their corresponding tokens q_i, d_j, respectively. Both sets of weights must be determined consistently, as they define the measure used to calculate the EMD.

The weights can reflect different aspects of their tokens. For instance, they can be the TF-IDF scores of the tokens within a given data set. However, because TF-IDF scores depend on the data set the model is applied to, they cannot be measured for out-of-vocabulary tokens, i.e., tokens not present in the training set. In addition, under-represented tokens, which are rare in the training set, might get a TF-IDF score that wrongly reflects the token's importance due to its lack of presence in the data set.

To facilitate handling both out-of-vocabulary and under-represented tokens, we set the weights to be uniformly distributed, i.e. $w_{q_i} = \frac{1}{n}$, $i = 1, \ldots, n$ and $w_{d_j} = \frac{1}{m}$, $j = 1, \ldots, m$. The uniform distribution will give equal importance to their respective query and document tokens, including to those that are out-of-vocabulary and under-represented,

which allows the algorithm to take into consideration all of the provided tokens. Note that the algorithm permits non-uniform weight distribution as long as the weight constraints stated in Equation (4.2) are met.

Using distance matrix D and token distributions μ_q, ν_d, we calculate the EMD between the query and document by finding the matrix $P \in \mathbb{R}_{\geq 0}^{n \times m}$ that solves the following optimization problem:

$$\begin{aligned} \text{minimize } \mathrm{EMD}(q,d) &= \sum_{i=1}^{n}\sum_{j=1}^{m} P_{i,j} D_{i,j}, \\ \text{subject to } \sum_{j=1}^{m} P_{i,j} &\leq w_{q_i}, \ \forall i \in \{1,\dots,n\}, \\ \sum_{i=1}^{n} P_{i,j} &\leq w_{d_j}, \ \forall j \in \{1,\dots,m\}, \\ \sum_{i=1}^{n}\sum_{j=1}^{m} P_{i,j} &= 1 = \min\Big\{\sum_{i=1}^{n} w_{q_i}, \sum_{j=1}^{m} w_{d_j}\Big\}. \end{aligned} \tag{4.3}$$

Note that Equation (4.3) is the optimal transport problem formulation in the discrete case for the cost matrix $C \in \mathbb{R}^{n \times m}$ with the entries $C_{i,j} = D_{i,j}$ given by Equation (4.1). The solution P is called the *transportation matrix* and gives the optimal way of transporting the mass from the distribution associated with the query to the distribution associated with the document. Because the distributions correspond to tokens, the transportation matrix can be viewed as the best way of moving the mass between the query tokens and the document tokens w.r.t. token distance matrix D.

After calculating all the document relevance scores, the model assigns the document ranks; documents with lower EMD scores are assigned a higher rank. The model then sorts the documents based on their rank and outputs the final order. Figure 4.2 shows the overall document ranking steps.

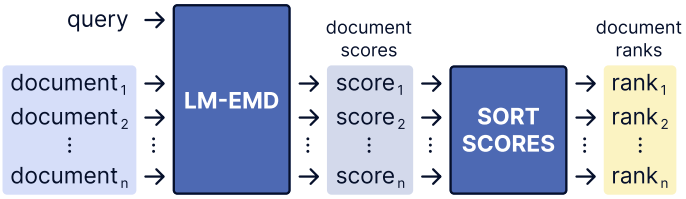


Figure 4.2: The LM-EMD document ranking steps. The document's rank is assigned based on the EMD score given by the LM-EMD model.

Interpreting document relevance

In addition to calculating the document's relevance score, the distance matrix and transportation matrix provide insight into why such a score was assigned to the document.

The solution of the optimal transport problem given by Equation (4.3) can be interpreted by looking at the transportation matrix P entries. Because of the problem's constraints, matrix P is non-zero and has at least one positive entry. The entries of P show how the mass between the query and document tokens was transported. For instance, the entry $P_{i,j}$ shows the amount of mass that is moved between query token q_i and document token d_j; the smaller the entry's value, the smaller the mass moved between the tokens. In order to reach the minimum of the optimal transport problem, the matrix P must have larger values at entries associated with the tokens that are closer in the embedding space

(entry $P_{i,j}$ must be large when distance $D_{i,j}$ is small). Otherwise, it would transport mass between tokens that are further apart and hence have a higher transport cost. Thus, the matrix could not be the solution to the optimization problem. Furthermore, the larger the number of query tokens and document tokens close in the embedding space, the smaller the EMD score - making the document more relevant to the query.

To view which document tokens have the largest significance on the relevance score, we must only read the values of the transportation matrix. Figure 4.3 shows an illustrative example of interpreting the token importance using the similarity matrix and transportation matrix. The similarity matrix contains various degrees of similarity between the tokens. The majority are distant, but there are a few that are close, e.g., the token pair (q_4, d_5) exhibits strong similarity. This is then reflected in the transportation matrix which indicates a strong match between the pair, hence token d_5 is significant for the document's relevance. Similarly, we find that (q_3, d_4) also have a strong match. While the (q_1, d_7) and (q_2, d_8) pairs have a medium match, the remainder only have a weak match. Hence, we conclude that the document is relevant because its tokens d_4, d_5, d_7 and d_8 have a significant similarity to the query tokens.

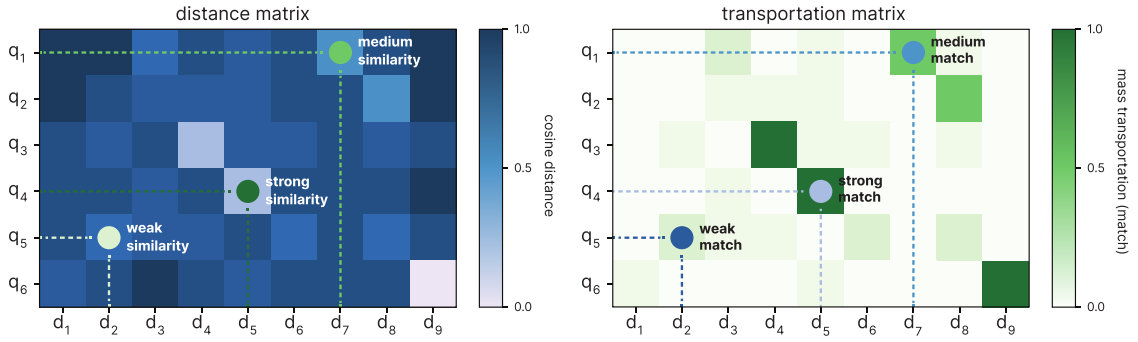


Figure 4.3: Interpreting the LM-EMD model output of a single document for a given query using the distance (left) and transportation (right) matrix. The colors in the matrices represent the intensity of the token distance (left) and their match (right), respectively.

4.1.3 Implementation details

In this section, we present how the LM-EMD model is developed. It is implemented using PyTorch [195], and its source code is available on GitHub [196].

Language model

The approach uses multilingual BERT, specifically the `bert-base-multilingual-cased` model[1], available via the HuggingFace's transformer library [197]. The language model consists of 110M parameters and was pre-trained on the top 104 languages, with the largest Wikipedia using the masked language modeling objective. The multilingual BERT uses the WordPiece tokenizer [198], which adds special word boundary symbols such that the original word sequence can be retrieved from the wordpieces sequence without ambiguity.

Similarity measure

The Earth Mover's Distance is calculated using the Sinkhorn-Knopp algorithm [189], an iterative algorithm for solving regularized EMD. The algorithm requires two hyper-

[1] `https://huggingface.co/bert-base-multilingual-cased`

parameters: (1) the number of iterations nit, and (2) regularization factor γ. The regularization factor is used to regulate the size of the regularization term and, thus, the precision of the final EMD optimization solution. Smaller factors provide more precise solutions but also introduce a higher risk of under- or over-flowing gradient values.

To provide the Sinkhorn-Knopp algorithm with enough iterations to converge to a solution, we set $nit = 500$. To analyze the impact of the regularization factor on the model's performance, we train three models using $\gamma = \{0.1, 1, 10\}$, respectively. These values are frequently used with the algorithm in other related works [189].

Model fine-tuning

The LM-EMD model parameters are updated using the AdamW optimizer [199] with the initial learning rate $= 10^{-6}$, epsilon $= 10^{-6}$, and weight decay $= 0.01$. These hyper-parameters were selected using grid search and monitoring the loss values of the first 1000 examples from the EN $\rightarrow$ DE training data set.

The model fine-tuning is performed on the selected language pair training sets for 1 epoch. We employ gradient accumulation and update the model after every 16th mini-batch, which consists of the query and its 40 relevant and irrelevant documents. We use one of the two loss functions: (1) cross-entropy, and (2) pairwise ranking loss function, defined as

$$L\big(q, d^+, d^-\big) = \max\left\{0, S(q, d^-) - S(q, d^+)\right\},$$

where S is the scoring function (i.e. the EMD), q is the query, and d^+, d^- are the relevant and irrelevant documents, respectively. The model's fine-tuning and evaluation were performed on a Tesla V100 PCIe 32 GB graphic card.

4.1.4 Experimental setting

We now present the experiment's setting. We introduce the data sets and how they are prepared for the experiments. Next, we describe the implementation details. Finally, we present the comparing methods used in the evaluation.

Data set

While the monolingual document retrieval task has a variety of open data sets that can be used for reproducible experiments, e.g. Reuters Corpora [200], MS MARCO [201], and TREC [202] among others, open cross-lingual data sets are all derived from Wikipedia. These include:

- WikiCLIR [203]. A large-scale data set consisting of German queries and English Wikipedia articles as documents.
- Large-Scale CLIR Dataset [204]. A data set derived from Wikipedia is comprised of over 2.8 million English queries and relevant documents from 25 selected languages.
- CLIRMatrix [205]. A massive collection of bilingual and multilingual data sets. They collected 49M unique queries and 34B (query, document, label) triplets.
- WikiMatrix [206]. The data set contains 135M parallel sentences from 1.6k different language pairs in 85 languages.

We employ the Large-Scale CLIR and the CLIRMatrix data sets to evaluate our model. One could argue that the data sets are unsuitable for evaluating multilingual language

models since most are trained using Wikipedia. Nevertheless, we chose them to achieve reproducible results and ensure a healthy data set size.

Each row in the data set contains the following values:

- *Query.* The English query is used for finding relevant documents.
- *Document.* The text of the document in the foreign language.
- *Relevance.* The relevance score between the query and the document.

While the training set is structured such that for each query, there are 4 irrelevant documents and 1 relevant document, the test set contains 39 irrelevant and 1 relevant document per query.

Data preparation

Large-Scale CLIR Dataset. We focus on the following language pairs: English (EN) → German (DE), French (FR), Tagalog (TL), Japanese (JA), and Swahili (SW), respectively. The languages were selected to evaluate the models on the same language family (EN, DE), different language families (FR, TL, SW, JA), low-resource languages (TL, SW), and a language that uses non-Latin script (JA).

In these data sets, the query is the first sentence of the English Wikipedia article with the article's title removed. Each document is limited to the first 200 words of the associated Wikipedia article. For each data set, we retrieve all queries, documents, and relevance scores and remove any trailing blank spaces. We scale the relevancy score values to 1 (relevant) and 0 (irrelevant). Table 4.1 shows the statistics of the selected language pairs within the Large-Scale CLIR data set.

Table 4.1: The Large-Scale CLIR data set statistics. The ratio represents the number of relevant and irrelevant documents per query. While the German, French, and Japanese language pairs are larger, the Tagalog and Swahili language pairs contain significantly fewer queries and documents, which is associated with their respective Wikipedia size.

	Training set			Test set		
Language pair	# queries	# docs	ratio	# queries	# docs	ratio
EN → DE	323,167	1,615,835	1/4	42,021	1,680,840	1/39
EN → FR	379,811	1,899,055	1/4	54,196	2,167,840	1/39
EN → JA	148,526	742,630	1/4	21,149	845,960	1/39
EN → TL	16,631	83,155	1/4	2,359	94,360	1/39
EN → SW	7,929	39,645	1/4	1,160	46,400	1/39

CLIRMatrix. With this data set, we focus on the language pairs English (EN) ↛ German (DE), French (FR), Slovene (SL), Croatian (HR), Serbian (SR), and Czech (CS), respectively. The languages were selected to evaluate the models on the same language family (EN, DE) and different language families containing mostly Slavic languages (FR, SL, HR, SR, CS), where one of them uses non-Latin script (SR).

In these data sets, the query text is generally a title of English Wikipedia articles, while the documents are the first 200 words of the associated Wikipedia article. We retrieve the queries, documents, and relevance scores and remove trailing blank spaces.

The possible relevance scores vary between 0 (irrelevant) and 6 (highly relevant). When creating the train and test set, we retrieve the most relevant document and a number of irrelevant documents to match the relevant/irrelevant ratio of the Large-Scale CLIR data set. Finally, we scale the relevancy score values to 1 (relevant) and 0 (irrelevant). Table 4.2 shows the statistics of the selected language pairs within CLIRMatrix.

Table 4.2: The CLIRMatrix Dataset statistics. The ratio represents the number of relevant and irrelevant documents per query. Due to the CLIRMatrix creation design, the language pairs have similar numbers of present queries and documents.

	Training set			Test set		
Language pair	# queries	# docs	ratio	# queries	# docs	ratio
EN → DE	10,000	50,000	1/4	1,000	40,000	1/39
EN → FR	10,000	50,000	1/4	1,000	40,000	1/39
EN → SL	9,999	49,995	1/4	1,000	40,000	1/39
EN → HR	10,000	50,000	1/4	999	39,960	1/39
EN → SR	9,994	49,970	1/4	1,000	40,000	1/39
EN → CS	9,998	49,990	1/4	999	39,960	1/39

Evaluation metrics

Following previous work [204], we use the P@1 (precision at 1) and MAP (mean average precision) metrics to measure the performance of the models.

The P@1 measure. This statistic measures if a relevant document is found at rank 1. It is a special case of the P@k (precision at k), which corresponds to the number of relevant documents among the retrieved ones:

$$\text{P@k} = \frac{|\{\text{retrieved documents}\} \cap \{\text{relevant documents}\}|}{|\{\text{retrieved documents}\}|} \in [0, 1].$$

The MAP score. This is the mean of the average precision scores for each query $q \in Q$. The average precision score is defined as

$$\text{AveP}(q) = \frac{1}{R} \sum_{k=1}^{n} \text{P@k} \cdot \text{rel@k},$$

where R is the total number of relevant documents, n is the number of retrieved documents, and rel@k signifies if the document at rank k is relevant or not:

$$\text{rel@k} = \begin{cases} 1, & \text{the document at rank } k \text{ is relevant,} \\ 0, & \text{the document at rank } k \text{ is irrelevant.} \end{cases}$$

The mean average precision is then given by equation:

$$\text{MAP} = \frac{\sum_{q \in Q} \text{AveP}(q)}{|Q|} \in [0, 1].$$

Models with higher MAP scores assign higher ranks to relevant documents.

Comparing models

In this section, we present the comparing models used to compare with the proposed LM-EMD. The models are selected to have an attribute similar to the proposed model, which is the possibility of using document representations generated ahead of time. Models that require both the query and document texts at inference time for constructing the input in the form of `[CLS] query [SEP] document [SEP]`, such as those presented in some of the related work [80]–[82], cause massive computational overhead on larger data sets and are thus not considered in the evaluation.

NRM. The Neural Ranking Model [204] uses word embeddings and convolutional layers to generate the query and document vector representations. It was evaluated separately on the Large-Scale CLIR Dataset's high-resource and low-resource language pairs. Since the model is already evaluated on the language pairs we use in the experiment, we only report their best performances from the paper.

WE. These models use aligned word embeddings [9] to create the document vector representation. Following previous work [75], [207], each query and document is represented as a weighted sum of its corresponding word embeddings:

$$\vec{v}_d = \sum_{t \epsilon T_d} w_t \vec{x}_t,$$

where T_d is the set of the unique terms in document d, and w_t and $\vec{x}_t$ are the weight and word embedding associated with term t, respectively. The similarity of two vectors is then measured with the cosine distance, where the smaller distance signifies greater similarity. We compare three different weighted sums where the weight w_t equals to:

(i) *TF:* The *term frequency* of term t in document d: $\mathrm{TF}(t, d) = f_{t,d}$, where $f_{t,d}$ is the number of times term t occurs in document d.

(ii) *IDF:* The *inverse document frequency* of term t within training set D: $\mathrm{IDF}(t, D) = \log \frac{|D|}{1+|\{d \epsilon D : t \epsilon d\}|}$.

(iii) *TFIDF:* The product of the *term frequency and inverse document frequency* of term t in document d and training set D: $\mathrm{TFIDF}(t, d, D) = \mathrm{TF}(t, d) \cdot \mathrm{IDF}(t, D)$.

The queries are represented by the TF-weighted sum vectors. The embeddings used by the models are open source [208].

WE-EMD. The model uses the same aligned word embeddings as the WE models and EMD to calculate the similarity between the query and documents [62]. The EMD is solved using the Entropic regularized optimal transport [63] available in the POT Python library [209].

BERT. The models are similar to SBERT [84], where the query and document representation vectors are generated separately using the same multilingual BERT. The inputs are formatted to `[CLS] input [SEP]`. We compare three different vector representations:

(i) *CLS:* The vector associated to the special `[CLS]` token.

(ii) *MAX:* The positional maximum values across the output token vectors.

(iii) *MEAN:* The mean of the output token vectors.

The document relevance is measured using the cosine distance, where the smaller distance signifies greater similarity. The models are fine-tuned using the same hyper-parameters described in Section 4.1.3. In addition, we train the models using both cross-entropy and pairwise ranking loss to analyze their impact on the models' performance.

PSOSL. This end-to-end robust framework combines the pre-trained Polyglot [210] embeddings with the smooth cosine similarity measure and the smooth ordinal search loss objective function [211]. The model was evaluated on the Large-Scale CLIR Dataset, specifically on two high-resource languages (French and Italian) and two low-resource languages (Swahili and Tagalog). Although the model's authors added 40 additional irrelevant documents for each query, changing the amount of data used in the experiments, we report their best performances from the paper.

4.1.5 Experiment results

Performance on Large-Scale CLIR data set. Table 4.3 shows the experiment results of the best-performing models on the language pair data sets. Because aligned word embeddings were unavailable for the JA and SW languages, we omitted the corresponding WE model performances from the results. In addition, because the original paper did not evaluate the PSOSL model for the DE and JA languages, and due to our limited resources for performing the experiments, their associated performances are not included. The reported LM-EMD is trained with pairwise ranking loss and regularization factor $\gamma = 0.1$.

Table 4.3: The LM-EMD experiment results on Large-Scale CLIR data set (P@1 and MAP scores reported). The bold and underlined values represent the best and second-best performances on the given language pair, respectively.

	P@1					MAP				
Model	EN → DE	EN → FR	EN → TL	EN → JA	EN → SW	EN → DE	EN → FR	EN → TL	EN → JA	EN → SW
NRM	0.710	0.760	0.570	0.730	0.600	0.820	0.850	0.690	0.840	0.730
PSOSL	-	0.438	0.596	-	0.600	-	0.841	0.772	-	0.776
WE-TF	0.737	0.718	0.590	-	-	0.836	0.823	0.704	-	-
WE-IDF	0.744	0.718	0.594	-	-	0.840	0.822	0.712	-	-
WE-TFIDF	0.764	0.754	0.597	-	-	0.854	0.846	0.710	-	-
WE-EMD	0.867	0.828	0.606	-	-	0.919	0.895	0.723	-	-
BERT-CLS	**0.978**	**0.978**	**0.851**	**0.955**	**0.913**	**0.987**	**0.987**	**0.912**	**0.973**	**0.947**
BERT-MAX	0.941	0.948	0.798	0.912	0.824	0.964	0.969	0.874	0.946	0.886
BERT-MEAN	0.967	0.958	0.786	0.941	0.835	0.980	0.976	0.874	0.965	0.897
LM-EMD	0.977	0.974	0.801	**0.955**	0.890	0.986	0.985	0.874	**0.973**	0.932

The BERT-CLS model performs best across all data sets, followed by the proposed LM-EMD, which is on par with the high-resource languages. A significant decrease in the performance of all models is seen in the low-resource languages, especially on EN → TL. This is accredited to the limited resources the low-resource languages have for training. Overall, LM-EMD outperforms BERT-MAX and BERT-MEAN, both using all token embeddings to measure the document's relevance.

Performance on CLIRMatrix. Given the results of the Large-Scale CLIR data set, we focus on the best-performing BERT-based models and LM-EMD when comparing the performance using the CLIRMatrix data set. The results are presented in Table 4.4, where we report the LM-EMD model trained using the pairwise ranking loss and a regularization factor of $\gamma = 0.1$.

Table 4.4: The LM-EMD experiment results on the CLIRMatrix data set (P@1 and MAP scores reported). The bold and underlined values represent the best and second-best performances on the given language pair, respectively.

	P@1						MAP					
Model	EN → DE	EN → FR	EN → SL	EN → HR	EN → SR	EN → CS	EN → DE	EN → FR	EN → SL	EN → HR	EN → SR	EN → CS
BERT-CLS	0.876	0.887	0.720	0.714	0.747	**0.787**	0.918	0.920	0.812	0.794	0.829	0.852
BERT-MAX	0.849	0.847	0.674	0.661	0.694	0.719	0.896	0.889	0.772	0.751	0.788	0.800
BERT-MEAN	**0.886**	**0.891**	**0.756**	**0.737**	**0.767**	0.783	0.923	0.922	**0.832**	**0.815**	**0.840**	**0.855**
LM-EMD	0.885	0.890	0.698	0.698	0.713	0.776	**0.924**	**0.923**	0.796	0.790	0.803	0.845

The results indicate that BERT-MEAN performs the best across almost all data sets. The performance of all models is lower in all Slavic languages compared to the EN → DE and FR language pairs. Additionally, the overall performance of the models is lower on the CLIRMatrix data set compared to the Large-Scale CLIR data set. This may be related to the query length, as CLIRMatrix queries are shorter Wikipedia article titles, unlike the longer queries in the Large-Scale CLIR data set comprised of the first sentence of the queries Wikipedia article.

While the BERT-CLS, BERT-MAX, and BERT-MEAN models return a single value representing the document's relevance, the LM-EMD outputs values that can be used to identify why a document is relevant. This makes the LM-EMD model superior in the sense of informativeness; in addition to measuring relevance, it can also determine the tokens that make the document relevant. The following section describes how the LM-EMD model identifies these tokens.

Interpretability

This section describes how the LM-EMD model scores can be interpreted. We show that the document's relevance is associated with the number and intensity of the matches between the query's and document's tokens. In addition, the relevance can be described graphically.

We prepared the query "Who was the first president of the United States?" and five documents in German: one contains the answer (George Washington), one is similar but incorrect (Abraham Lincoln), and three irrelevant documents with various degrees of similarity. Using the LM-EMD model we calculate all document relevance scores with respect to the query. The scores are shown in Table 4.5.

Table 4.5: The relevance score of the German documents for the query "Who was the first president of the United States?" A smaller score means a greater relevance.

Question: Who was the first president of the United States?

Document	Score
George Washington war von 1789 bis 1797 der erste Präsident der Vereinigten Staaten von Amerika.	0.6332
Abraham Lincoln amtierte von 1861 bis 1865 als 16. Präsident der Vereinigten Staaten von Amerika.	0.6773
Christoph Kolumbus wurde der erste Vizekönig der las Indias genannten Gebiete.	0.7392
Augusta Ada King-Noel, Countess of Lovelace, allgemein als Ada Lovelace bekannt war eine britische Mathematikerin.	0.7871
Marie Skłodowska Curie war eine Physikerin und Chemikerin polnischer Herkunft, die in Frankreich lebte und wirkte.	0.7926

The model correctly detected that George Washington was the most relevant document, followed by Abraham Lincoln. To identify which tokens contributed the most to their respective score, we follow the interpretation described in Section 4.1.2 and generate the graphs of the distance matrix and transportation matrix values for each document (similar to Figure 4.3). The graphs are shown in Figure 4.4.

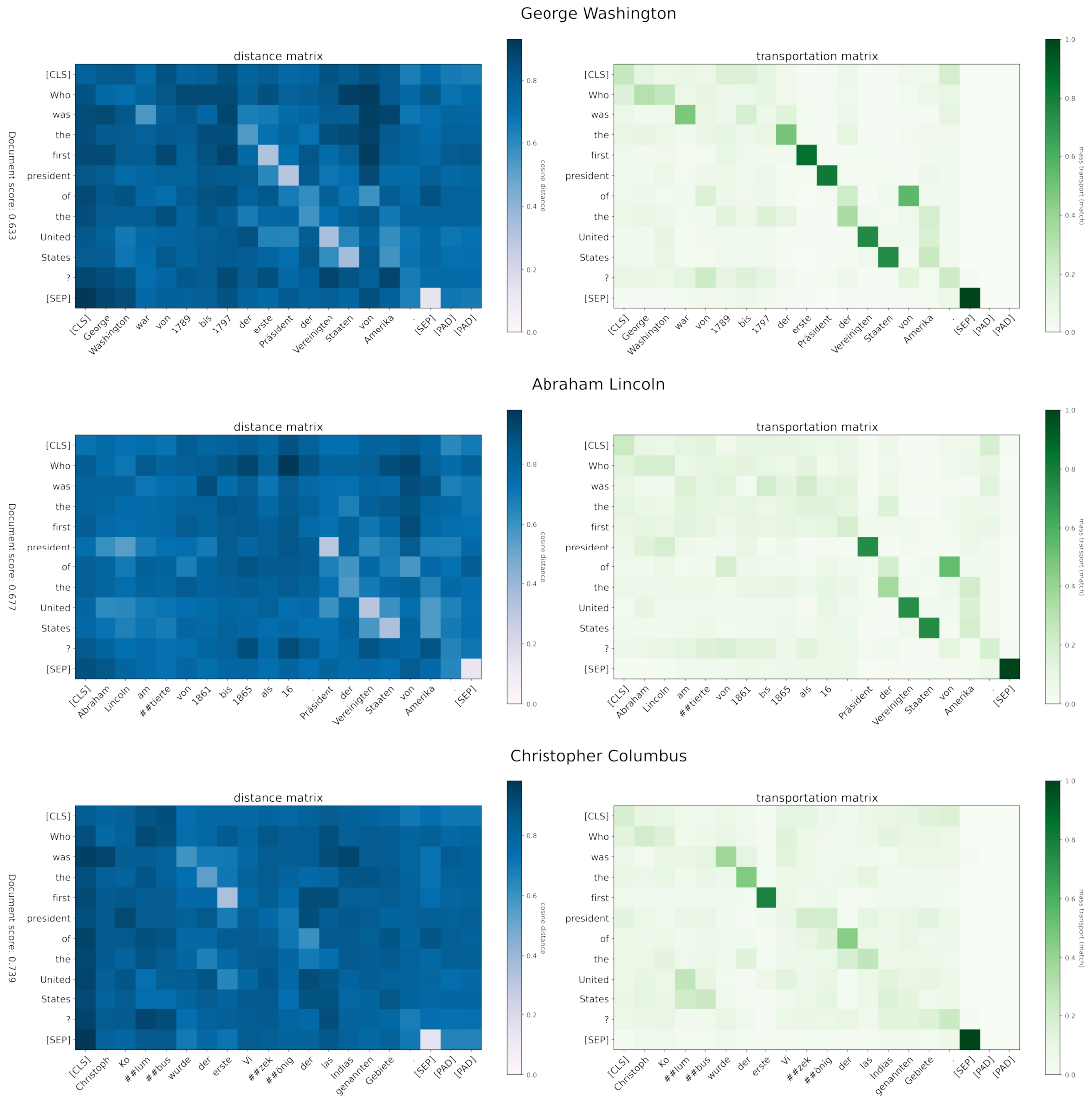


Figure 4.4: The distance (left) and transportation (right) matrix of the first three documents (George Washington, Abraham Lincoln, and Christopher Columbus) from the example. The `[PAD]` tokens are ignored when calculating the EMD score.

The George Washington graphs (first row) show that almost all query tokens strongly match those in the document. Most are literal translations (e.g. “president”, “Präsident”), but some are matched due to their semantic relatedness, such as the query word “Who” and document words “George Washington”.

The Abraham Lincoln graphs (second row) show that only half of the query tokens have a strong match. This is due to the smaller number of close tokens in the distance matrix. Because of this, the document got a worse relevance score than the George Washington sentence.

In comparison, the third set of graphs belongs to the Christopher Columbus document, which is irrelevant. The graphs show the model finds a strong match only for the literal translation of “first” and some conjunction words, such as “was”, “the”, and “of”. Surprisingly, it also finds a weak match between the words “president” and “Vizekönig” (en. viceroy), which are positions of power, and between “United States” and “Kolum-

bus"[2]. This shows the model can detect the slightest relations between the cross-lingual tokens. Nevertheless, the document has a small number of matches and thus has a smaller relevance score.

As seen in the example, the document's relevance is associated with the number of strong matches between the query's and document's tokens; the greater the number and the stronger the match between the tokens, the greater the relevance. Because of this, the LM-EMD model generates interpretable relevance scores supported by the graphic visualizations, as shown in this section.

Parameter analysis

We compare multiple LM-EMD models using different regularization factor values used by the Sinkhorn-Knopp algorithm. To assess the impact of the loss function on the model's performance, we trained models using the cross-entropy (CE) and pairwise ranking (PR) loss, respectively. The comparison was performed on the Large-Scale CLIR data set with the results shown in Table 4.6.

Table 4.6: The performance comparison of the LM-EMD model trained with different regularization factor (γ) values, and using the pairwise ranking (PR) or cross-entropy (CE) loss function during training. The bold value highlights the biggest difference in performance (Δ) on the language pair.

		P@1					MAP				
Param	Loss	EN → DE	EN → FR	EN → TL	EN → JA	EN → SW	EN → DE	EN → FR	EN → TL	EN → JA	EN → SW
$\gamma = 0.1$	PR	0.977	0.974	0.801	0.955	0.890	0.986	0.985	0.874	0.973	0.932
	CE	0.876	0.843	0.674	0.846	0.631	0.927	0.909	0.793	0.908	0.754
	Δ	**0.101**	**0.131**	0.127	**0.109**	**0.259**	**0.059**	**0.076**	0.081	**0.065**	**0.178**
$\gamma = 1$	PR	0.970	0.968	0.809	0.910	0.859	0.982	0.981	0.883	0.946	0.913
	CE	0.876	0.846	0.669	0.846	0.617	0.927	0.910	0.790	0.907	0.747
	Δ	0.094	0.122	**0.140**	0.064	0.242	0.055	0.071	**0.093**	0.039	0.177
$\gamma = 10$	PR	0.965	0.961	0.805	0.899	0.835	0.979	0.978	0.881	0.941	0.899
	CE	0.878	0.846	0.671	0.848	0.628	0.928	0.911	0.792	0.909	0.753
	Δ	0.087	0.115	0.134	0.051	0.207	0.051	0.067	0.089	0.032	0.146

The models using PR outperform the models trained with CE, showing that PR matches better with the document retrieval evaluation metrics. The performance of all models trained with PR decreases with increasing regularization factor, except for the EN → TL language pair. The biggest change is seen in the JA and SW languages. While the decrease in JA can be because of the difference between the scripts used to represent the languages, the behavior of the SW can be accredited to its polysynthetic language, e.g. having morphologically complex words with a large number of affixes which EN does not have. In comparison, when trained with CE, the model's performance first decreases and then increases on low-resource languages. At the same time, the performance of the high-resource ones is best when using the largest regularization factor.

A similar analysis, comparing the cross-entropy (CE) and pairwise ranking loss (PR), was done on BERT-based models, which gave similar results. Due to our limited resources for performing the experiments, we only report the results on EN → DE, FR, and TL language pairs. The analysis results are shown in Table 4.7.

[2]A quick search shows that there are multiple cities in the United States that are named Columbus, after Christopher Columbus.

Table 4.7: The comparison of BERT-{CLS, MAX, MEAN} performances using the pairwise ranking (PR) loss and cross-entropy (CE) functions during training, with the difference in performance (Δ).

		P@1			MAP		
Model	Loss	EN → DE	EN → FR	EN → TL	EN → DE	EN → FR	EN → TL
BERT-CLS	PR	0.978	0.978	0.851	0.987	0.987	0.912
	CE	0.893	0.855	0.729	0.937	0.916	0.838
	Δ	0.085	0.123	0.122	0.050	0.071	0.074
BERT-MAX	PR	0.941	0.948	0.798	0.964	0.969	0.874
	CE	0.861	0.824	0.676	0.918	0.898	0.802
	Δ	0.080	0.124	0.122	0.046	0.071	0.072
BERT-MEAN	PR	0.967	0.958	0.786	0.980	0.976	0.874
	CE	0.861	0.824	0.676	0.918	0.898	0.802
	Δ	0.106	0.134	0.110	0.062	0.078	0.072

As presented in this section, optimal transport, in combination with language models, can be used to measure document relevancy. Relevancy is assessed by how close the document words are to the query words in the embedding space. However, the Earth Mover's Distance overlooks sentence structure when measuring the proximity of two texts. The following section will address this issue by incorporating sentence structure into our similarity measure.

4.2 Including Sentence Structure into Similarity Measure

Text generation models are becoming more present in our everyday lives. These models, capable of facilitating tasks such as text translation, summarization, transcription generation, and question answering, are employed to optimize and automate our work processes. However, the effectiveness of these models depends on their ability to produce informative, structurally, and grammatically accurate text. Thus, evaluating these models before using them is essential to ensure their usefulness.

Using automatic evaluation metrics has proven to be a valuable tool in developing and evaluating text-generation models. In the past, the performance of such models was measured through manual scoring by human evaluators, which was a time-consuming and costly process. To mitigate these limitations, automatic evaluation metrics were developed to assess the generated text's quality accurately while reducing the time and cost associated with manual scoring. Since then, several such metrics have been proposed, each with its own technique for evaluating the quality of the generated text. These metrics have allowed for a more efficient and cost-effective evaluation of text generation models.

Metrics for assessing the performance of text generation models often overlook different evaluation criteria. Typically, the metrics are optimized to assign a single overall quality score, which is often associated with the adequacy of the generated text - if it accurately represents all of the required information. However, the fluency of the generated text is another important criterion that should be considered; fluency refers to the natural-sounding and grammatical correctness of the text, as well as its coherence and well-structuredness. To ensure a comprehensive evaluation of text generation models, it is necessary to consider both the adequacy and fluency of the generated text.

The main aim of this work is to develop a metric for measuring the performance of text generation models which considers both the adequacy and fluency of the generated text.

The main research objectives are to (1) evaluate the metric's adequacy-based performance on different language pair data sets consisting of the generated and reference texts and (2) determine the sensitivity of the metric to fluency-related modifications in sentences.

We propose a text generation metric named *Order-Preserving Wasserstein Score* (OPWScore), which uses language models to generate the token's semantic representations and a word order-preserving optimal transport to include both adequacy and fluency of the generated text when assessing its quality. The metric is evaluated on two human-annotated translation data sets consisting of twenty language pairs drawn from six distinct languages belonging to various language groups and using different writing scripts. The evaluation is performed using Pearson's r and Kendall's τ correlation coefficients. The results show that the proposed metric performs slightly worse than the best-performing comparing metrics but is most sensitive to the word order permutations among the embedding-based metrics.

We also analyzed how sensitive the metric is to modifications affecting sentence fluency. A statistical measure was developed to quantify the average change in the metrics' score for various modifications. The results show that the proposed OPWScore exhibits the highest sensitivity to fluency-related modifications among all embedding-based metrics.

The remaining sections are structured as follows. We first describe the metric architecture in Section 4.2.1 and present its implementation details in Section 4.2.2. We then present the experimental setting in which we evaluate the metric in Section 4.2.3. Finally, we describe the experiment results in Section 4.2.4.

4.2.1 The OPWScore metric architecture

This section proposes a text generation metric named Order-Preserving Wasserstein Score, abbreviated as OPWScore, that considers both the adequacy and fluency of the sentence. It is a modification of the LM-EMD model presented in Section 4.1, where EMD is replaced with the word order-preserving OT used to calculate the final score. The optimal transport theory used to create this metric is presented in Section 3, while its extension on how it considers the word order is found in Section 3.2. The metric's architecture is depicted in Figure 4.5. The implementation details are presented in Section 4.2.2.

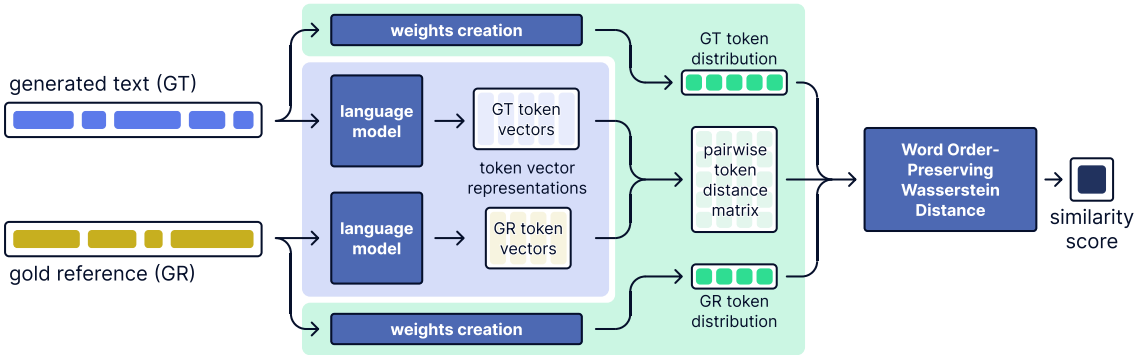


Figure 4.5: The OPWScore metric architecture. The architecture includes generating token vector representations, which are then used to solve the word order-preserving OT problem and calculate the similarity score between the generated text and its gold reference.

Token vector representations

We use language models to extract the current contextual semantic meaning of the text. This is done by splitting the text into subword segments (tokens) using the tokenizer

associated with the model, a subword segmentation algorithm that converts the text into character n-grams. The model then generates the vector representations (embeddings) for each token based on its context, capturing its semantic meaning. As a result, tokens with similar semantic meanings are closer in the embedding space.

Let g and r be the generated text and its associated gold reference, respectively. Using a language model, we extract their corresponding tokens and embeddings. Since the language model has a limit to its input length, we split longer texts into multiple subtexts so that each subtext can be processed. The subtext tokens and embeddings are then concatenated together for further processing.

When a word is split into multiple tokens, we treat each as a separate sentence component; an approach also used in other embedding-based evaluation metrics [102], [103]. This will be alleviated when calculating order-preserving Wasserstein distance, which maintains the global order of tokens. This means tokens are matched with those of similar indices, thereby somewhat restricting the matching of token affixes with non-affixes (see Figure 4.6).

The tokens and their embeddings are then joined and ordered to create the sequence sets S_g and S_r of the generated text g and the gold reference r. The sequence sets consist of (token, embedding) pairs used for measuring the quality of the generated text:

$$S_g = \left[(g_i, \vec{v}_{g_i})\right]_{i=1}^{n}, \quad S_r = \left[(r_j, \vec{v}_{r_j})\right]_{j=1}^{m},$$

where n, m are the number of extracted tokens, g_i, r_j are the tokens extracted from the generated text g and reference r, and $\vec{v}_{g_i}, \vec{v}_{r_j}$ are the contextual embeddings of their corresponding tokens, respectively. Although the generated text and gold reference can share the same tokens, their associated embeddings can differ since they reflect the token's context and semantic meaning.

Measuring the quality of the generated text

The order-preserving Wasserstein distance (OPW) measures the distance between two sequences and their assigned probability measures. We define the quality of the generated text as the OPW value between the text and its associated reference. A smaller OPW implies the generated text is better in terms of its grammar and contextual meaning of words. Let $D \in \mathbb{R}^{n \times m}$ be the matrix of distances between the query and document tokens. Similar as in Section 4.1.2, matrix D contains the cosine distances between the token embeddings:

$$D_{i,j} = d_{\cos}(g_i, r_j) = 1 - \frac{\langle \vec{v}_{g_i}, \vec{v}_{r_j} \rangle}{\|\vec{v}_{g_i}\|_2 \|\vec{v}_{r_j}\|_2} \geq 0. \tag{4.4}$$

Let μ_g and ν_r be the token distributions of the generated text and gold reference:

$$\begin{aligned} \mu_g &= \sum_{i=1}^{n} w_{g_i} \delta_{g_i}, \quad \sum_{i=1}^{n} w_{g_i} = 1, \\ \nu_r &= \sum_{j=1}^{m} w_{r_j} \delta_{r_j}, \quad \sum_{j=1}^{m} w_{r_j} = 1, \end{aligned} \tag{4.5}$$

where δ_{g_i}, δ_{r_j} are the Dirac delta functions as defined in Equation (3.7), and the weights w_{g_i}, w_{r_j} represent the mass of their corresponding tokens g_i, r_j, respectively.

Depending on how they are formulated, the weights can reflect different aspects of their tokens. One option is to set the weights to be uniformly distributed, i.e. $w_{g_i} = \frac{1}{n}$, $i = 1, \ldots, n$ and $w_{r_j} = \frac{1}{m}$, $j = 1, \ldots, m$. The uniform distribution will give equal importance to their respective tokens. Another option is to set the weights to reflect the token's presence in a data set. In such cases, the TF-IDF would be an appropriate option; the TF-IDF represent

the importance of a token within a data set. It combines the token frequency (TF) [212] within a text and its inverse document frequency (IDF) [213], which measures how unique a token is within the whole data set. However, the TF-IDF values depend on the data set to which it is applied, and they cannot be measured for out-of-vocabulary tokens, i.e., tokens not present in the used data set. In addition, when assigning weight distributions to short texts, such as single sentences, it can be assumed that the token's frequency within a sentence will be around one, thus the TF value can be omitted and only the IDF of the token is used as the weight. Note that the algorithm permits other weight distributions as long as the weight constraints stated in Equation (4.5) are met.

Using distance matrix D and token distributions μ_g, ν_r, we calculate the OPW between the generated text and gold reference by finding the matrix solution $P \in \mathbb{R}_{\geq 0}^{n \times m}$ of the following optimization problem:

$$\begin{aligned} \text{minimize } \mathrm{OPW}(g,r) &= \sum_{i=1}^{n}\sum_{j=1}^{m}\left(P_{i,j}D_{i,j} - \lambda_1 \frac{P_{ij}}{\left(\frac{i}{n}-\frac{j}{m}\right)^2+1}\right), \\ \text{subject to } &\sum_{j=1}^{m} P_{i,j} \leq w_{g_i},\ \forall i \in \{1,\dots,n\}, \\ &\sum_{i=1}^{n} P_{i,j} \leq w_{r_j},\ \forall j \in \{1,\dots,m\}, \\ &\sum_{i=1}^{n}\sum_{j=1}^{m} P_{i,j} = 1 = \min\left\{\sum_{i=1}^{n} w_{g_i}, \sum_{j=1}^{m} w_{r_j}\right\}, \end{aligned} \tag{4.6}$$

where $\lambda_1 \in \mathbb{R}_{\geq 0}$ is the regularization parameter that determines the importance of local homogeneity of the solution P. Note that Equation (4.6) is the order-preserving optimal transport problem formulation in the discrete case for cost matrix $C \in \mathbb{R}^{n \times m}$ with entries $C_{i,j} = D_{i,j}$ given by Equation (4.4). In addition, when calculating the approximate solution of the above problem with the Sinkhorn-Knopp algorithm, we include the Kullback-Leibler divergence for prior distribution G defined in Equation (3.10). The solution is then used to measure the quality of generated text g based on its reference r.

4.2.2 Implementation details

In this section, we describe the details of the OPWScore development. The metric is implemented using PyTorch [195], and its source code is available on GitHub [214].

Language model

Depending on the target language, the metric uses one of the two pre-trained language models: fine-tuned RoBERTa and multilingual BERT. Both language models are available via the HuggingFace's transformer library [197].

If the target language is English, the metric uses the fine-tuned RoBERTa model [17] fine-tuned on the Multi-Genre Natural Language Inference corpus [215], more precisely the `roberta-large-mnli` model[3]. In addition, related work [103] shows using different layer outputs yields different performance scores. Based on their analysis, we decided to use the embeddings returned by the 19th layer as they give the best performance for the BERTScore metric. This also enables us to perform a more aligned comparison with the compared metrics.

[3] `https://huggingface.co/roberta-large-mnli`

When the target language is non-English, the metric uses multilingual BERT [20], more specifically `bert-base-multilingual-cased` model[4]. The language model was pre-trained on the top 104 languages with the largest Wikipedia using the masked language modelling objective. With similar reasoning as before, we use the embeddings returned by the 9th layer of the model since it was reported it performs best for the task we are exploring in this paper [103].

Weight distribution

To calculate the token weights for a particular target language, we use the gold references from the WMT17 metric evaluation data set [216] to calculate the token's inverse document frequency (IDF). Since all text in the data sets is in the form of single sentences, we omit calculating the token frequency (TF) as we assume it will be close to one. We split the sentences into tokens using the corresponding language model's tokenization method (depending on the target language) and calculate the IDF value. At evaluation time, we use the token's corresponding IDF to construct the sentence's distribution. If the token was previously unseen, i.e. was not present in the WMT17 data set, we set its IDF to a small value (i.e. 10^{-5}) to ensure a stable calculation of the OPW. To analyze the impact of the weight distribution on the metric's performance, we evaluate two metrics using the uniform and the IDF weight distributions, respectively, and report our results in Section 4.2.4, Hyper-parameter analysis.

Quality measure

The OPW is calculated using the Sinkhorn-Knopp algorithm [63], an iterative algorithm for solving regularized OT. Depending on the model, the algorithm requires the following two hyper-parameters: (1) the number of iterations *nit*, and (2) the two regularization parameters λ_1 and λ_2 from Equation (3.11). The regularization factors are used to regulate the size of the regularization term and, thus, the precision of the final OPW value. While λ_1 corresponds to the inverse difference moment as defined in Equation (3.9), λ_2 corresponds to the Kullback-Leibler divergence used in Equation (3.11).

To provide the Sinkhorn-Knopp algorithm with enough iterations to converge to a solution, we set $nit = 100$. To analyze the impact of the regularization factors on the metric's performance, we evaluate nine metrics using $\lambda_1 \in \{1, 0.1, 0.01\}$ and $\lambda_2 \in \{0.1, 1, 10\}$ and report our findings in Section 4.2.4, Hyper-parameter analysis.

4.2.3 Experimental setting

This section introduces the data sets used and how the data is prepared for the experiments. Next, we present the evaluation metrics, followed by the descriptions of the baseline metrics. Finally, we describe the implementation details of the proposed metric.

Data set

To evaluate text generation metrics on different languages, one has a selection of automatic translation data sets to choose from [216]–[220]. To evaluate our metric, we use the WMT18 [218] and WMT20 [219] metric evaluation data sets, both containing generated outputs of translation systems across several language pairs, gold standard references, and human judgment scores.

[4] `https://huggingface.co/bert-base-multilingual-cased`

The source sentences and reference translations are taken from a news corpus with a various number of sentences for each translation direction; the translations are from English (EN) to Czech (CS), German (DE), Finnish (FI), Russian (RU), Turkish (TR), and Chinese (ZH), and from the same set of languages to English. The languages come from five different language families and use four distinct language scripts.

The human judgment scores are direct assessments between the gold reference and the generated text. The scores are collected by human evaluators with various expertise. While the data set contains two types of human scores, we only use the segment-level scores, i.e., values between 0 and 100 assigned to the generated text based on its ability to express the meaning of its corresponding gold reference adequately. Because of this, the human scores are biased towards assessing the adequacy of the generated texts rather than their fluency. We chose the data sets because they are popular for evaluating text generation metrics and enable us to achieve reproducible results.

Data preparation

For each available language pair, we prepare the set of rows containing (1) the **source** sentence in the source language, (2) the single **gold reference** in the target language, (3) the **translation** generated from the source sentence, and (4) the **segment-level score** that the human evaluators assigned to the translation. We do not perform any text preprocessing on any of the sentences. Table 4.8 shows the statistics of the prepared WMT18 and WMT20 language pair subsets. For instance, in WMT18, we have 7,629 examples translated from English to Czech and 8,732 from Czech to English.

Table 4.8: The WMT18 and WMT20 language pair set statistics. For each language pair, the left number is the number of from-English translation examples, and the right is the number of to-English translation examples. The language pairs FI ↔ EN and TR ↔ EN are not available in the WMT20 data set.

Data set	CS ↔ EN	DE ↔ EN	FI ↔ EN	RU ↔ EN	TR ↔ EN	ZH ↔ EN
WMT18	7,629 / 8,732	10,208 / 28,404	8,097 / 14,965	16,748 / 13,157	3,132 / 12,851	22,128 / 25,352
WMT20	14,543 / 7,510	9,693 / 8,529	- / -	11,188 / 9,664	- / -	14,975 / 27,329

Evaluation measures

Following previous works [97]–[100], [102], [103], we use Pearson's r and Kendall's τ to measure the metric's performance.

Pearson correlation coefficient. The sample Pearson's r measures the linear correlation between two sample sets X and Y. It is defined as the ratio between the covariance of the two samples and the product of their standard deviations.

$$r_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} \in [-1, 1],$$

where $\operatorname{cov}(X,Y)$ is the covariance estimate between X and Y, and σ_X, σ_Y are standard deviation estimates of sample sets X and Y, respectively. The $r_{X,Y}$ value is between -1 and 1, indicating the capacity in which the sample sets X and Y are linearly correlated.

Kendall rank correlation coefficient. The Kendall's τ is a measure of rank correlation based on the similarity of the orderings of the data points between two variables X and Y. It

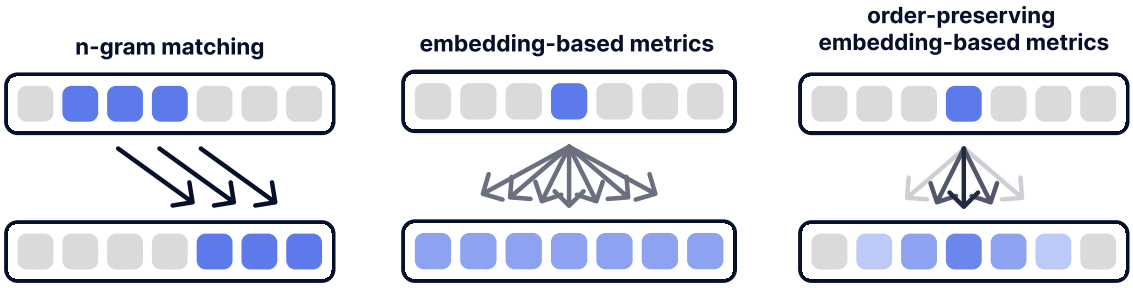
is often used as a test statistic in a statistical hypothesis test to determine whether two variables may be regarded as statistically dependent. The coefficient is defined as

$$\tau = \frac{Concordant - Discordant}{Concordant + Discordant} \in [-1, 1],$$

where *Concordant* is the number of times a metric assigns a higher score to the "better" generated text and *Discordant* is the number of times a metric assigns a higher score to the "worse" generated text or the scores assigned to both texts are the same. The τ value ranges between -1 and 1, where -1 indicates perfect disagreement between the two rankings, 0 suggests that the variables are independent and non-constant, and 1 indicates there is perfect agreement between the two rankings.

Baseline metrics

This section presents the baseline metrics used to compare with the proposed OPWScore metric. The metrics were selected to enable quality assessment on cross-lingual data sets without additional training or fine-tuning. Metrics that support only mono-lingual text comparison [101], [105] are thus not considered in the evaluation. The final selection includes both n-gram matching approaches and embedding-based metrics. Figure 4.6 shows the differences between the selected approaches.



(a) The n-gram matching approaches try to match sequences of words between texts.

(b) These metrics match tokens based on their embeddings but do not consider word order.

(c) These metrics match tokens based on their embedding but are restricted to tokens with similar positions. OPWScore falls in this group.

Figure 4.6: The different approaches evaluating text generation methods.

BLEU. This metric is one of the first to be developed for automatically evaluating machine translation models [97]. It measures the quality of the candidate sentence by comparing them to their reference sentences. The comparison is made by measuring the *modified n-gram precision*, an advanced method of counting the number of n subsequent words (called n-grams) that appear in both the candidate and each reference sentence. It returns a score between zero and one, where higher scores mean more sentence overlap. As stated in its associated paper, a translation using the same words (1-grams) as in the references tends to satisfy adequacy, while the longer n-gram matches account for fluency. In the experiments, we measure the BLEU score for $n \in \{1, 2, 3, 4\}$ and denote them as BLEU-1, BLEU-2, BLEU-3, and BLEU-4, respectively.

ROUGE-L. The ROUGE package contains various metrics for evaluating the quality of generated texts by using n-grams, word sequences, and word pairs between the candidate

and reference sentences [98]. The metric used in the experiment is ROUGE-L which measures the longest common sub-sequence in the candidate and reference sentences. The ROUGE-L score is between zero and one, where higher values mean better matching between the sentences.

METEOR. This metric is based on a generalized concept of unigram matching between the candidate and reference translations [99]. The matching is performed on the surface form (exact matching of unigrams), the stemmed forms (matching words that have the same word stem), and the word's meaning (matching words if they are synonyms of each other). It then uses a combination of precision and recall of the unigram matching and a measure involving the longest n-grams common to the reference and candidate sentences to calculate the METEOR score. The score is between zero and one, where the higher values mean a better matching between the sentences.

BERTScore. This embedding-based metric uses contextual embeddings to represent the tokens in the reference and candidate sentence and perform token matching based on the token's pairwise cosine similarities [103]. The contextual embeddings are taken from the language model's layer, which provides the best abstraction of the token, as stated in the metric's corresponding paper. The final score is an F1 measure computed from the precision and recall of matches between the tokens in the candidate and reference sentence. The metric also uses the IDF scores of tokens as weights, denoting their importance in the sentences. In the experiments, we use the publicly available implementation of BERTScore[5]. Note that we use the metric's default settings and do not perform any fine-tuning.

MoverScore. This metric combines contextual embeddings of the tokens in the reference and candidate sentences with optimal transport to measure the quality of the generated text [102]. The optimal transport is measured using Euclidean distance on the embedding representations of the token n-grams. The n-gram representations are a weighted sum of the token's embeddings, where the weights are the IDF scores of their corresponding token in the corpus. The model can use the output representation of any layer within the chosen language model. In our experiments, the MoverScore metric is represented by the *LM-EMD* model, which is a MoverScore variation where it uses token unigrams (1-grams) and cosine distance instead of Euclidean distance to measure the distances between the token's embeddings (see Section 4.1.2).

COMET. This supervised metric uses a cross-lingual language model to create contextual embeddings for the source, candidate and reference tokens, which are then used to calculate the quality of the candidate text [104]. The quality calculation is done by training a regression model, combining and concatenating the embeddings and sending them through a feed-forward layer, which outputs the final score. Since the language model provides a contextual embedding for each layer, the metric was trained to combine the outputs of all layers to generate the optimal token embedding for the task. In the experiments, we use the publicly available implementation of COMET[6]. Note that COMET's default settings are fine-tuned for the WMT20 data set, thus it is expected to have a higher performance score.

[5] `https://github.com/Tiiiger/bert_score`
[6] `https://unbabel.github.io/COMET/html/index.html`

4.2.4 Experiment results

This section shows the experiment's results. We report the analysis results of the metric's sensitivity to fluency-based modifications, followed by the adequacy-based performance analysis on the two data sets and the metric's hyper-parameter analysis.

Sensitivity to fluency-based modifications

Human judgement scores of the modified sentences would be required to evaluate the metric's sensitivity to fluency-based modifications. Since access to such data sets is unavailable, we resolved to measure the change in metric scores when a fluency-based modification is performed on a sentence. In this analysis, we focus on the sensitivity of metrics to change in word order. We define a statistic for measuring the word order sensitivity of a metric as follows:

$$\mathrm{WOS}(M|S) = \frac{1}{|S|}\sum_{s_i \in S}\left|M(s_i, s_i) - \hat{s}_i\right|, \quad \hat{s}_i = \frac{1}{k}\sum_{j=1}^{k} M\big(p_j(s_i), s_i\big), \tag{4.7}$$

where M is the metric, $S = \{s_i\}_{i=1}^{\ell}$ is the set of sentences, $p_j(s_i)$ is the sentence generated by modifying the word order of the sentence s_i using the modification function p_j, and ℓ and k are the number of sentences in the set and the number of distinct modification functions, respectively. The WOS statistic is a positive number that shows the average difference of metric scores when given a perfect sentence s_i and a sentence created by modifying the word order in the perfect sentence $p_j(s_i)$. Thus, the statistic measures how sensitive a metric is to word order, where higher values signify higher sensitivity.

We used the gold references taken from the WMT18 data set to evaluate the metrics. The used modification functions p_j shuffle the words in the sentences, following the directions from previous work [106]. We calculate each language pair's average word order sensitivity and report the results in Table 4.9. Note that we compare only unsupervised metrics; thus, we omit COMET from this analysis as it is a supervised regression metric.

Table 4.9: The scores of the fluency statistic tests on the WMT18 data set (WOS reported). The bold and underlined values represent the embedding-based metrics that are the most and second-most sensitive to word order permutations on a given language pair. The reported OPWScore uses IDF weights and regularization factors $\lambda_1 = 0.1$ and $\lambda_2 = 0.1$.

	Metric	CS → EN	DE → EN	FI → EN	RU → EN	TR → EN	ZH → EN	EN → CS	EN → DE	EN → FI	EN → RU	EN → TR	EN → ZH
n-gram matching	BLEU-1	0.046	0.045	0.048	0.060	0.054	0.056	0.022	0.050	0.048	0.041	0.009	0.756
	BLEU-2	0.927	0.931	0.930	0.934	0.925	0.093	0.920	0.927	0.914	0.924	0.900	0.168
	BLEU-3	0.991	0.993	0.993	0.994	0.988	0.993	0.989	0.990	0.987	0.991	0.972	0.157
	BLEU-4	0.996	0.998	0.998	0.998	0.994	0.996	0.996	0.994	0.995	0.995	0.977	0.096
	METEOR	0.536	0.538	0.534	0.540	0.541	0.536	0.535	0.549	0.542	0.557	0.532	0.458
	ROUGE-L	0.624	0.635	0.626	0.642	0.628	0.653	0.568	0.613	0.552	0.032	0.545	0.023
embed-based	BERTScore	0.144	0.147	0.143	0.148	0.144	0.149	0.230	0.273	0.201	0.242	**0.355**	0.019
	MoverScore	0.260	0.262	0.257	0.263	0.257	0.261	0.224	0.264	0.196	0.235	0.203	0.027
	OPWScore	**0.306**	**0.310**	**0.303**	**0.312**	**0.304**	**0.315**	**0.359**	**0.373**	**0.343**	**0.367**	0.345	**0.063**

Among the embedding-based metrics, OPWScore shows the highest sensitivity to word order across all language pairs, except for EN → TR, followed by MoverScore. Both metrics use optimal transport as part of their calculations. BERTScore is the least sensitive among the embedding-based metrics. Note that BERTScore and MoverScore calculate the scores in a way that is invariant to word order. However, the reported WOS results for these two metrics show some sensitivity. We accredit this to the used language models, which include the token's positional encodings when generating the token embeddings. Thus, the

final token representation might also include information about the token's position in the sentence.

The n-gram matching metrics are more sensitive to word order because they match exact sequences of words, i.e., the same words in the same sequence. In contrast, embedding-based metrics match words based on their contextual embeddings, making them less sensitive to word order. However, these metrics are more effective than n-gram matching ones in measuring text adequacy, as discussed in the following section.

Adequacy-based performance analysis

The adequacy-based performance experiments measure the correlation between the metrics' and human judgment scores between the generated translations and the gold references. The intention is to measure how well the generated translation captures the actual message of the source text. Based on the analysis results in Section 4.2.4, the reported OPWScore uses the IDF weights and the regularization factors $\lambda_1 = 0.1$ and $\lambda_2 = 0.1$.

WMT18 data set. The experiment results performed on the WMT18 language pairs are found in Table 4.10 and Table 4.11. The results show that COMET outperforms all other metrics in terms of performance. Among the unsupervised metrics, BERTScore is the best-performing one, followed by MoverScore, which is on par with most language pairs. OPWScore has a lower performance compared to the leading unsupervised embedding-based metrics, with an average decrease of 0.042 Pearson's r and 0.024 Kendall's τ points on to-English data sets and an average decrease of 0.07 Pearson's r and 0.048 Kendall's τ points on from-English data sets. However, OPWScore performs better than all of the n-gram matching metrics used in the evaluation.

Table 4.10: Metric performances on the WMT18 to-English translations (Pearson's r and Kendall's τ reported). The bold and underlined values represent the best and second-best performances among unsupervised metrics on a given language pair, respectively.

		Pearson's r							Kendall's τ						
	Metric	CS → EN	DE → EN	FI → EN	RU → EN	TR → EN	ZH → EN	Avg	CS → EN	DE → EN	FI → EN	RU → EN	TR → EN	ZH → EN	Avg
	SUPERVISED														
	COMET	0.550	0.632	0.526	0.499	0.516	0.464	0.531	0.380	0.452	0.353	0.338	0.374	0.314	0.369
	UNSUPERVISED														
n-gram matching	BLEU-1	0.208	0.368	0.220	0.235	0.271	0.250	0.259	0.149	0.258	0.152	0.167	0.196	0.169	0.182
	BLEU-2	0.240	0.390	0.225	0.246	0.261	0.255	0.270	0.162	0.274	0.155	0.174	0.194	0.178	0.190
	BLEU-3	0.233	0.385	0.197	0.230	0.236	0.230	0.252	0.157	0.261	0.140	0.163	0.173	0.166	0.177
	BLEU-4	0.216	0.335	0.172	0.210	0.213	0.204	0.225	0.147	0.248	0.132	0.152	0.158	0.156	0.166
	METEOR	0.271	0.405	0.241	0.263	0.285	0.273	0.290	0.186	0.284	0.165	0.184	0.210	0.187	0.203
	ROUGE-L	0.279	0.424	0.251	0.268	0.305	0.286	0.302	0.192	0.297	0.172	0.191	0.227	0.198	0.213
embed-based	BERTScore	**0.408**	**0.550**	0.369	**0.384**	0.403	**0.366**	**0.413**	**0.285**	**0.388**	**0.252**	**0.270**	0.296	**0.252**	**0.291**
	MoverScore	0.399	0.543	**0.375**	0.372	**0.409**	0.362	0.410	0.274	0.380	0.251	0.259	**0.300**	0.249	0.286
	OPWScore	0.382	0.517	0.332	0.340	0.346	0.307	0.371	0.266	0.368	0.226	0.245	0.270	0.225	0.267

WMT20 data set. Table 4.12 and Table 4.13 show the experimental results performed on the WMT20 data sets. Similar to before, COMET is, on average, the best-performing metric across all language pairs. When comparing the results of the English data sets, all three unsupervised embedding-based metrics have similar performances concerning Kendall's τ coefficient. When comparing the results on non-English data sets, BERTScore performs best, followed by MoverScore. OPWScore has a lower performance compared to BERTScore, with an average decrease of 0.065 Pearson's r points and 0.028 Kendall's τ points, but performs better than all n-gram matching metrics across all data sets.

Table 4.11: Metric performances on the WMT18 from-English translations (Pearson's r and Kendall's τ reported). The bold and underlined values represent the best and second-best performances among unsupervised metrics on a given language pair, respectively.

		Pearson's r							Kendall's τ						
	Metric	EN → CS	EN → DE	EN → FI	EN → RU	EN → TR	EN → ZH	Avg	EN → CS	EN → DE	EN → FI	EN → RU	EN → TR	EN → ZH	Avg
	SUPERVISED														
	COMET	0.730	0.749	0.691	0.578	0.677	0.525	0.658	0.554	0.560	0.524	0.426	0.506	0.357	0.488
	UNSUPERVISED														
n-gram matching	BLEU-1	0.408	0.435	0.339	0.340	0.339	0.023	0.314	0.278	0.299	0.277	0.236	0.238	0.007	0.214
	BLEU-2	0.387	0.446	0.314	0.348	0.331	0.009	0.306	0.256	0.311	0.210	0.238	0.230	0.005	0.208
	BLEU-3	0.350	0.413	0.283	0.323	0.303	0.005	0.280	0.228	0.291	0.198	0.222	0.210	0.002	0.192
	BLEU-4	0.309	0.376	0.240	0.292	0.257	0.000	0.246	0.207	0.272	0.178	0.207	0.187	0.004	0.176
	METEOR	0.398	0.456	0.327	0.368	0.359	0.026	0.322	0.265	0.318	0.218	0.253	0.250	0.019	0.221
	ROUGE-L	0.422	0.456	0.341	0.044	0.403	0.037	0.284	0.288	0.317	0.227	0.028	0.287	0.014	0.194
embed-based	BERTScore	**0.501**	**0.567**	0.405	**0.420**	**0.535**	**0.394**	**0.470**	**0.348**	**0.394**	0.271	**0.290**	**0.384**	**0.275**	**0.327**
	MoverScore	0.500	0.550	**0.409**	0.417	0.440	0.369	0.448	0.345	0.384	**0.272**	0.287	0.316	0.258	0.310
	OPWScore	0.451	0.491	0.339	0.361	0.415	0.342	0.400	0.307	0.349	0.222	0.251	0.297	0.248	0.279

Table 4.12: Metric performances on the WMT20 to-English translations (Pearson's r and Kendall's τ reported). The bold and underlined values represent the best and second-best performances among unsupervised metrics on a given language pair, respectively.

		Pearson's r					Kendall's τ				
	Metric	CS → EN	DE → EN	RU → EN	ZH → EN	Avg	CS → EN	DE → EN	RU → EN	ZH → EN	Avg
	SUPERVISED										
	COMET	0.183	0.637	0.267	0.278	0.341	0.122	0.274	0.164	0.192	0.188
	UNSUPERVISED										
n-gram matching	BLEU-1	0.113	0.528	0.163	0.186	0.248	0.062	0.213	0.112	0.131	0.130
	BLEU-2	0.133	0.452	0.156	0.198	0.235	0.081	0.213	0.113	0.144	0.138
	BLEU-3	0.123	0.370	0.136	0.183	0.203	0.074	0.205	0.099	0.138	0.129
	BLEU-4	0.109	0.304	0.121	0.166	0.175	0.065	0.193	0.088	0.129	0.119
	METEOR	0.160	0.549	0.168	0.218	0.274	0.104	0.249	0.112	0.157	0.156
	ROUGE-L	0.155	0.566	0.175	0.225	0.280	0.099	0.252	0.119	0.162	0.158
embed-based	BERTScore	**0.196**	0.605	**0.246**	0.264	0.328	0.130	0.261	**0.159**	**0.185**	**0.184**
	MoverScore	0.187	**0.628**	0.244	**0.267**	**0.332**	0.124	**0.262**	0.158	**0.185**	0.182
	OPWScore	0.184	0.567	0.223	0.236	0.303	**0.132**	**0.262**	0.158	0.178	0.183

While BERTScore and MoverScore demonstrate improved performance, they have limited capacity to consider the sentence's word order when calculating their scores, which can affect their perceived performance. In contrast, OPWScore addresses this issue by combining the semantic meaning of tokens through their vector representations and enforcing matching between tokens in corresponding positions via order-preserving optimal transport.

Hyper-parameter analysis

Weight distribution analysis. A comparison was performed between OPWScore using a uniform weight distribution and a distribution based on the IDFs of tokens calculated from the WMT17 data sets to evaluate the impact of different weight distributions on the metric's performance. Table 4.14 and Table 4.15 show the OPWScore performance comparison when using either the uniform or IDF weights. The remaining hyper-parameters were fixed using the regularization factors $\lambda_1 = 1$ and $\lambda_2 = 0.1$. The evaluation was performed on the WMT18 data sets.

The analysis shows that the metric performs better when using IDF weights across all

Table 4.13: Metric performances on the WMT20 from-English translations (Pearson's r and Kendall's τ reported). The bold and underlined values represent the best and second-best performances among unsupervised metrics on a given language pair, respectively.

		Pearson's r					Kendall's τ				
	Metric	EN → CS	EN → DE	EN → RU	EN → ZH	Avg	EN → CS	EN → DE	EN → RU	EN → ZH	Avg
	SUPERVISED										
	COMET	0.614	0.496	0.390	0.274	0.444	0.384	0.263	0.282	0.247	0.294
	UNSUPERVISED										
n-gram matching	BLEU-1	0.340	0.240	0.168	0.015	0.191	0.215	0.121	0.123	0.019	0.120
	BLEU-2	0.341	0.226	0.169	0.003	0.185	0.219	0.126	0.120	0.014	0.120
	BLEU-3	0.303	0.205	0.157	0.001	0.167	0.196	0.119	0.107	0.014	0.109
	BLEU-4	0.270	0.182	0.145	0.002	0.150	0.178	0.109	0.094	0.010	0.098
	METEOR	0.352	0.260	0.172	0.006	0.198	0.224	0.136	0.122	0.011	0.123
	ROUGE-L	0.349	0.272	0.031	0.032	0.171	0.213	0.145	0.033	0.013	0.101
embed-based	BERTScore	**0.431**	**0.366**	**0.238**	**0.362**	**0.349**	**0.263**	**0.189**	**0.173**	**0.223**	**0.212**
	MoverScore	0.411	0.358	0.232	0.348	0.337	0.251	0.184	0.168	0.216	0.205
	OPWScore	0.355	0.311	0.207	0.268	0.285	0.226	0.175	0.154	0.179	0.184

Table 4.14: Analysis of using IDF and uniform weight distribution on the WMT18 to-English translations (Pearson's r and Kendall's τ reported). The bold values represent the best-performing performances on a given language pair.

	Pearson's r							Kendall's τ						
weight	CS → EN	DE → EN	FI → EN	RU → EN	TR → EN	ZH → EN	Avg	CS → EN	DE → EN	FI → EN	RU → EN	TR → EN	ZH → EN	Avg
IDF	**0.373**	**0.508**	**0.321**	**0.330**	**0.335**	**0.298**	**0.361**	**0.259**	**0.363**	**0.220**	**0.239**	**0.262**	**0.219**	**0.260**
uniform	0.307	0.418	0.243	0.262	0.270	0.229	0.288	0.217	0.292	0.169	0.193	0.219	0.174	0.211
Δ	0.066	0.090	0.078	0.068	0.065	0.069	0.073	0.042	0.071	0.051	0.046	0.043	0.045	0.050

Table 4.15: Analysis of using uniform and IDF weight distribution on the WMT18 from-English translations (Pearson's r and Kendall's τ reported). The bold values represent the best-performing performances on a given language pair.

	Pearson's r							Kendall's τ						
weight	EN → CS	EN → DE	EN → FI	EN → RU	EN → TR	EN → ZH	Avg	EN → CS	EN → DE	EN → FI	EN → RU	EN → TR	EN → ZH	Avg
IDF	**0.440**	**0.478**	**0.331**	**0.354**	**0.411**	**0.339**	**0.392**	**0.299**	**0.341**	**0.216**	**0.246**	**0.294**	**0.246**	**0.274**
uniform	0.377	0.393	0.244	0.305	0.311	0.255	0.314	0.255	0.278	0.193	0.214	0.225	0.195	0.227
Δ	0.063	0.085	0.087	0.049	0.100	0.084	0.078	0.044	0.063	0.023	0.032	0.069	0.051	0.047

language pairs; tokens that appear less frequently in the applied data set have a higher IDF value and thus can be somewhat used to measure the token's importance in the data set. Furthermore, stopwords, i.e., words found in almost all sentences, have a smaller IDF value and, therefore, have lower importance when analyzing the text's meaning. This can then be used to indicate which tokens contribute to the increased adequacy of the generated text. Although the analysis was performed on fixed regularization factors, the difference between the results across all language pairs is substantial to conclude that using IDF weights is superior to using a uniform distribution to measure the adequacy of the generated text.

Regularization factor analysis. We analyze the impact regularization factors have on the metric's performance. This is done by running nine evaluations with different regularization factor values. The results performed on the WMT18 data sets are found in Table 4.16 and

Table 4.17. The metrics perform best when both regularization factors λ_1 and λ_2 are smaller. When fixing λ_2 to a particular value, the metric's performance is not significantly different when changing the λ_1 value. The biggest difference in performance happens when fixing λ_2 = 0.1 and decreasing λ_1 from 1 to 0.1. However, the greatest performance decrease occurs when the value of λ_2 increases. We also experimented using smaller values of both λ_1 and λ_2, but the calculation of the OPW became unstable.

Table 4.16: Analysis of using different hyper-parameter values on the WMT18 to-English translations (Pearson's r and Kendall's τ reported). The bold and underlined values represent the best and second-best performances on a given language pair, respectively.

Params		Pearson's r							Kendall's τ						
λ_1	λ_2	CS → EN	DE → EN	FI → EN	RU → EN	TR → EN	ZH → EN	Avg	CS → EN	DE → EN	FI → EN	RU → EN	TR → EN	ZH → EN	Avg
1	0.1	0.373	0.508	0.321	0.330	0.335	0.298	0.361	0.259	0.363	0.220	0.239	0.262	0.219	0.260
1	1	0.327	0.449	0.275	0.284	0.293	0.245	0.312	0.230	0.317	0.188	0.208	0.232	0.185	0.227
1	10	0.315	0.432	0.265	0.272	0.283	0.233	0.300	0.221	0.303	0.181	0.199	0.255	0.177	0.218
0.1	0.1	0.382	0.517	0.332	0.340	0.346	0.307	0.371	**0.266**	0.368	0.226	0.245	**0.270**	0.225	**0.267**
0.1	1	0.329	0.450	0.276	0.285	0.293	0.245	0.313	0.231	0.318	0.188	0.208	0.233	0.185	0.227
0.1	10	0.315	0.432	0.266	0.272	0.282	0.233	0.300	0.221	0.303	0.181	0.199	0.225	0.177	0.218
0.01	0.1	**0.383**	**0.518**	**0.333**	**0.341**	**0.347**	**0.308**	**0.372**	**0.266**	**0.369**	**0.227**	**0.246**	**0.270**	**0.226**	**0.267**
0.01	1	0.329	0.451	0.276	0.285	0.293	0.272	0.318	0.230	0.318	0.189	0.208	0.233	0.185	0.227
0.01	10	0.315	0.432	0.266	0.272	0.283	0.233	0.300	0.221	0.303	0.181	0.199	0.225	0.177	0.218

Table 4.17: Analysis of using different hyper-parameter values on the WMT18 from-English translations (Pearson's r and Kendall's τ reported). The bold and underlined values represent the best and second-best performances on a given language pair, respectively.

Params		Pearson's r							Kendall's τ						
λ_1	λ_2	EN → CS	EN → DE	EN → FI	EN → RU	EN → TR	EN → ZH	Avg	EN → CS	EN → DE	EN → FI	EN → RU	EN → TR	EN → ZH	Avg
1	0.1	0.440	0.478	0.331	0.354	0.411	0.339	0.392	0.299	0.341	0.216	0.246	0.294	0.246	0.274
1	1	0.396	0.420	0.273	0.318	0.346	0.279	0.339	0.268	0.298	0.178	0.222	0.246	0.209	0.237
1	10	0.386	0.405	0.262	0.310	0.333	0.268	0.327	0.261	0.287	0.172	0.217	0.238	0.202	0.230
0.1	0.1	0.451	0.491	0.339	0.361	0.415	**0.342**	0.400	0.307	**0.349**	0.222	0.251	**0.297**	**0.248**	0.279
0.1	1	0.397	0.422	0.273	0.319	0.347	0.280	0.340	0.268	0.299	0.179	0.223	0.247	0.210	0.238
0.1	10	0.386	0.405	0.262	0.310	0.333	0.268	0.327	0.261	0.287	0.172	0.217	0.238	0.202	0.230
0.01	0.1	**0.452**	**0.492**	**0.340**	**0.362**	**0.416**	**0.342**	**0.401**	**0.308**	**0.349**	**0.223**	**0.252**	**0.297**	**0.248**	**0.280**
0.01	1	0.397	0.422	0.273	0.319	0.347	0.280	0.340	0.268	0.299	0.179	0.223	0.247	0.210	0.238
0.01	10	0.386	0.405	0.262	0.310	0.333	0.268	0.327	0.261	0.287	0.172	0.217	0.238	0.202	0.230

4.3 Discussion

In this chapter, we highlight the advantages of employing sentence structure to access cross-lingual similarity in the information retrieval context and evaluate the content and grammar appropriateness of generated text. The LM-EMD model, applied to cross-lingual information retrieval, effectively retrieves pertinent information while offering insights into the reasons behind the relevance of the retrieved documents. Furthermore, the OPWScore metric is sensitive to word permutations, capturing nuances in grammar while maintaining the ability to measure content appropriateness. It is crucial to note that these models depend on the chosen text representation approach. The following discussion focuses on the findings concerning the LM-EMD and the OPWScore methods.

4.3.1 Discussion on LM-EMD results

In this section, we discuss the results of the LM-EMD experiment. We highlight the model's strengths, weaknesses, and theoretical and practical implications.

Model performance

When evaluating the Large-Scale CLIR data set, the proposed LM-EMD model shows comparable performance with the best-performing comparing model, BERT-CLS, on the high-resource languages, with an average of 0.968 P@1 and 0.981 MAP score. However, LM-EMD performance is significantly lower than that of BERT-CLS on low-resource languages, with an average of 0.846 P@1 and 0.903 MAP compared to the average of 0.882 P@1 and 0.930 MAP achieved by BERT-CLS.

When comparing LM-EMD to the BERT-MAX and BERT-MEAN, using all of the token embeddings to measure the document's relevance to the given query, the LM-EMD performs similar or better than the other two models across all language pairs. This shows that EMD is a superior measure as it considers all of the token distances when assigning the relevance score. This is also reflected when comparing WE-based methods where WE-EMD has the best performance scores.

In contrast, experiments on the CLIRMatrix data set show that the LM-EMD model performs generally worse than the BERT-CLS and BERT-MEAN models on Slavic languages. With an average of 0.721 P@1 and 0.809 MAP score, it performs, on average, about 0.040 P@1 and 0.030 MAP worse than the best-performing BERT-MEAN model. This shows that the LM-EMD model struggles when comparing English and Slavic languages. This issue may stem from the differences between the languages, which make it challenging for the language model to contextualize and compare texts in the embedding space.

Furthermore, when comparing the performance of models on both the Large-Scale CLIR and the CLIRmatrix EN → DE and FR language pairs, we find that the models generally perform worse on the CLIRmatrix dataset, which features shorter queries. This suggests that all approaches perform better with longer texts of similar lengths. This is understandable, as shorter query texts, which can have multiple meanings in different contexts, make retrieving relevant documents more challenging for the information retrieval model.

Model interpretability

Unlike the cosine distance, EMD can generate interpretable relevance scores. The EMD solution outputs the transportation matrix, indicating which query-document token pairs are semantically similar and thus improve the relevance score. The similarity it detects ranges from distant connections to synonyms and literal translations. However, EMD is more computationally expensive than cosine distance as it needs to run the Sinkhorn-Knopp algorithm to get the final score. It is also prone to introducing a higher risk of under- or over-flowing the gradient values when the regularization factor is too small. Because of this, the LM-EMD might be more appropriate to use "per request", e.g. when the user requests an explanation of why a document is relevant to their query.

Analysis of loss function selection

The analysis of the loss function's impact on the model's performance shows that models trained with pairwise ranking outperform those with cross-entropy. The pairwise ranking is more appropriate for document retrieval models because it assesses if the order of the retrieved documents is correct, unlike cross-entropy, which measures if the model can identify the document as relevant. While the Large-Scale CLIR Dataset has only one relevant document per query, other data sets might have multiple relevant documents, making pairwise ranking the more appropriate function.

4.3.2 Discussion on OPWScore results

In this section, we discuss the OPWScore experimental results. We touch upon the metric's advantages and disadvantages over the baseline metrics and their theoretical and practical implications.

Sensitivity to fluency-based modifications

The analysis shows OPWScore is the most sensitive to word order modifications compared to all other unsupervised embedding-based metrics. However, the results do not show how word order would affect the metric's correlation with the human judgement scores. For this, each modified sentence would have to be scored by human evaluators concerning the sentence's fluency, as suggested in a related work [106]. While this might be resolved in the future, most existing data sets do not contain such scores. To this end, statistics, such as word order sensitivity defined with Equation (4.7), could help understand the metric's capacity to include different aspects of the generated text. While the metrics do not necessarily return values on the same scale[7], the statistics can still provide some insight into how well does the metric handle different modifications of sentences.

Adequacy-based performance analysis

Analyzing the experimental results shows that the performance of the proposed OPWScore metric is lower than that of BERTScore, the best performing unsupervised embedding-based metric, for an average of 0.07 Pearson's r and 0.048 Kendall's τ points. However, the metric has comparable Kendall's τ scores to other embedding-based metrics on most WMT20 language pairs. In addition, the OPWScore's average Kendall's τ score is higher on from-English language pairs, showing that the metric is more appropriate for assessing the quality of non-English generated texts.

Hyper-parameter analysis

The results show that the OPWScore performs best when using IDF weights and smaller regularization factors λ_1 and λ_2. IDF weight distribution enables the metric to assign higher importance to tokens that are not present in more sentences within the data set. However, when a token is novel, i.e. unseen during the IDF calculation process, the IDF cannot correctly reflect the token's importance, which is necessary when assessing the adequacy of the generated text. This issue is not unique to OPWScore as other metrics, such as BERTScore and MoverScore, use IDF values to calculate their score. Research in alternative weight distributions reflecting the token's importance could improve all embedding-based text-generation metrics. Nonetheless, the IDF weights have improved the OPWScore's performance.

The regularization factors regulate the extent to which the metric considers the fluency aspects of the generated text. As described in Section 4.2.1, the regularization factors correspond to (1) the inverse difference moment, which tries to preserve the temporal position of the words associated with λ_1, and (2) the Kullback-Leibler divergence between the solution of the order-preserving optimal transport problem and a prior distribution, corresponding to λ_2. Both factors enforce the preservation of word order. However, they might be too strict since the message or information can be written in different ways under the constraints of the language. Potential research directions would be finding better prior distributions and other ways of considering the fluency of the generated text.

[7]The BLEU, METEOR, ROUGE-L, and BERTScore metrics return values between zero and one, while MoverScore and OPWScore return values greater than zero.

Chapter 5

Online Text Clustering with Neural Networks

This chapter focuses on research associated with online text clustering algorithms that use neural network models. Section 5.1 introduces a methodology for creating novel news clustering data sets. With it, we created a data set comprising of news reporting the 2021 Tokyo Olympics, intended for evaluating online news clustering algorithms. Section 5.2 proposes a novel two-stage online cross-lingual news clustering algorithm that uses unconventional distance metrics to cluster articles. Finally, Section 5.3 discusses the experiment results and the advantages and disadvantages of the work presented in this chapter.

The data set introduced in Section 5.1 is publicly available [221], [222] and presented in a scientific paper currently under review [223], while Section 5.2 is based on work that is also under review [224]. This chapter introduces three scientific contributions: [SC-2] a new temporal data set consisting of labeled multilingual news articles intended for evaluating news stream clustering approaches, [SC-3] a novel news stream clustering approach that uses Transformer-based language models and an unconventional method for measuring text similarity and [SC-5] evaluation of the proposed methods on various data sources and real-world scenarios.

5.1 Multilingual News Clustering Data Set Creation Process

Through news articles, we can learn about global events. Different publishers report the same event from various perspectives, highlighting what they find essential for their audience. Depending on the publisher's country, articles may be in different languages and may reflect the writer's biases. Thus, news articles are key for identifying world events, their coverage, and their global significance. To analyze these aspects, we need effective methods to group multilingual news articles based on their events, which typically involve similar entities (who/what was involved), time (when it happened), and place (where it happened).

There is a scarcity of data sets available for evaluating online multilingual news clustering algorithms. Most existing news datasets support research in classifying news articles into topics or domains. Some datasets [109], [110], [225], [226] contain articles annotated with a few labels, usually corresponding to the news domain (e.g. Sports, Business, Tech, World). Other datasets [112], [227]–[229], have annotated articles suitable for topic detection. Some recent datasets [230], [231] have multiple label groups corresponding to specific news aspects, such as political bias, reliability, and transparency. While they help classify news into topics or domains, these datasets are not necessarily suitable for identifying news

events, as the labels are often too broad and invariant to the temporal and geographical information of the article.

Furthermore, most datasets contain primarily English articles, making them unsuitable for multilingual tasks. Those that are multilingual are typically designed for other news-related tasks, such as discrimination [232] and alignment [233] between languages, language simplification [234], and news summarization [235]. Some multilingual news datasets [236], [237] are developed to analyze world events and cultural narratives, among others. In addition to datasets, news monitoring systems such as Event Registry [113] and the GDELT Project [115] track news reported from various international sources and perform data analysis. Both systems provide detailed event records, including date, location, and involving actors. While the GDELT Project provides event information, it does not include the articles used to detect the event. In contrast, the Event Registry provides news articles and event information, but the event clusters are typically monolingual. Our research identified only one multilingual news dataset with articles annotated based on the events they cover. This dataset was prepared by Miranda et al. [114] for evaluating news stream clustering algorithms. However, it contains a low number of events in the wide time range, so it may not be suitable for developing approaches focused on high-frequency events, i.e., events that happen close in temporal proximity and possibly in similar locations.

The manual creation of novel multilingual news clustering data sets is resource-intensive work. First, the articles must be collected from various sources and languages. Then, the articles must be analyzed and grouped depending on their content and publishing time. In addition, the annotator must speak various languages to understand the content of multilingual news articles. This requires expert linguists and translators to perform manual annotations, which is slow and time-consuming, especially when the data set requires many articles. Automating this process would speed up the data set creation and help the annotators with their work.

Recent advances in natural language processing have paved the way for automated methods that significantly reduce the efforts for manual data set creation. These methods use pre-trained language models to extract, categorize, and cluster multilingual news articles. In addition, automatic machine translation services enable employing annotators who are non-speakers of the selected languages. As a result, these solutions can accelerate the creation of relevant data sets by creating the initial clusters, which the annotators inspect and perform the necessary modifications. The final data set can be used to develop online news clustering algorithms further.

This section aims to develop a methodology for creating novel multilingual news clustering data sets by automatizing the news article collection and clustering process. This allows the annotators to focus on inspecting, joining, splitting, and removing articles from the created clusters.

The main research objectives are to (1) create a novel multilingual news clustering data set containing articles in languages that belong to different language families and are written in different scripts and (2) analyze and compare the data set with the existing multilingual news stream clustering data sets.

We focus on creating a multilingual data set of articles reporting the 2021 Tokyo Olympics, denoted as OG2021. The Olympics, spanning 18 days, presented a dense array of sub-events happening simultaneously, including articles with temporal, geographical, and contextual similarities that may challenge separation. The created data set contains 10,940 articles in nine different languages, labeled into 1,350 clusters. About 28% of clusters contain articles written in two or more languages. The articles are written in multiple languages and annotated based on the events they report; articles on the same event have the same annotation. Figure 5.1 illustrates the schematic overview of the novel OG2021

dataset preparation, including news article retrieval, annotation, and technical evaluation. The code used to create the data set is available on Github [238].

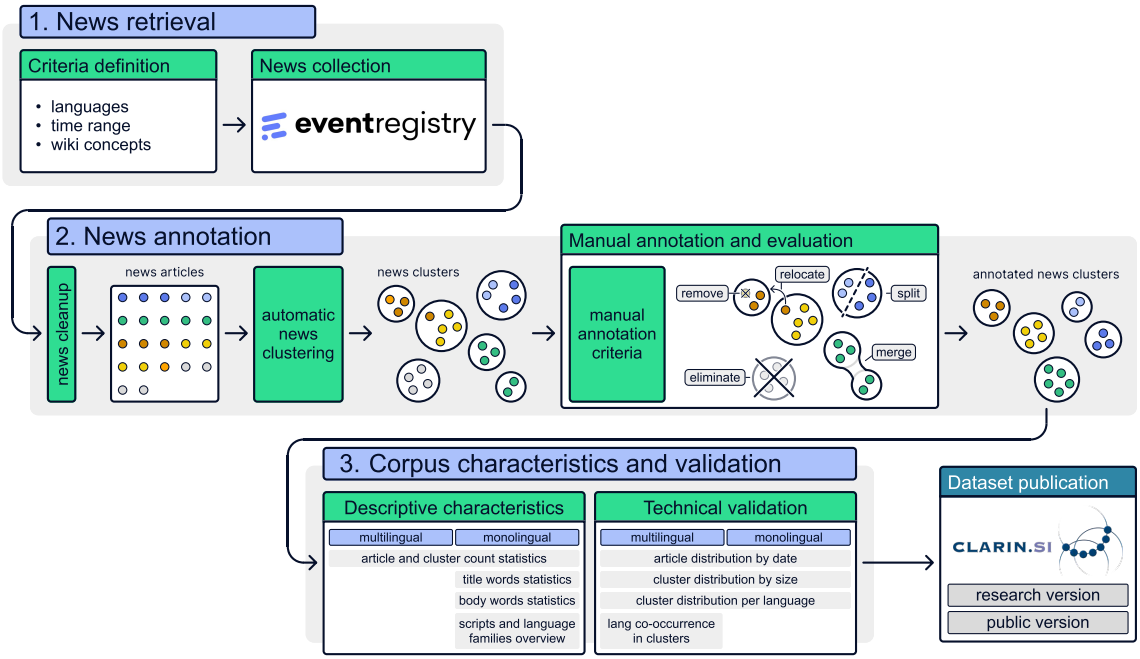


Figure 5.1: The schematic overview of the OG2021 development.

The remaining sections are structured as follows. We first describe the news article acquisition process in Section 5.1.1. Next, we present the data set creation process in Section 5.1.2, including automatic news cluster creation, manual inspection, and annotation. Various statistics and characteristics of the newly created OG2021 data set are presented in Section 5.1.3. Finally, we compare it with the existing data sets in Section 5.1.4.

5.1.1 News retrieval

This section describes the acquisition of the initial set of news articles focusing on the 2021 Tokyo Olympics. It includes the definition of the criteria and the retrieval process of relevant articles. It also presents the statistics of the data set obtained.

Criteria definition

To retrieve the appropriate articles, we first defined the criteria that the articles must follow. The requirements consist of three conditions: (1) the languages in which the article must be written, (2) the publication time, which must be within a predefined time range, and (3) the contextual concepts the article must include. Table 5.1 shows the overview of the defined conditions. Below is a detailed description of each criterion.

Table 5.1: The news retrieval criteria conditions.

No.	Condition type	Condition values
1	Languages	English, Portuguese, Spanish, French, Russian, German, Slovenian, Arabic, Chinese
2	Publication time range	July 1, 2021 - August 14, 2021
3	Contextual concepts	`Olympic Games`, `Japan` and at least one of `basketball`, `sports climbing`, `swimming`, `judo`, `rowing`, `skateboarding`, or `table tennis`

Languages. We aim to include articles in languages from diverse language families and scripts to represent a wide range of linguistic features. We chose English, Portuguese, Spanish, French, Russian, German, Slovenian, Arabic, and Chinese. The selected languages cover different language families (Germanic, Italic, Slavic, Semitic, and Sinitic) and are written in different scripts (Latin, Cyrillic, Arabic, and Chinese). We believe the languages provide sufficient diversity regarding language similarity and media coverage.

Publication time range. To capture relevant articles, we limited our selection to those published between July 1, 2021, and August 14, 2021. This time range includes articles from three weeks before the start of the 2021 Tokyo Olympics (July 23, 2021) to one week after its conclusion (August 8, 2021). This allowed us to gather articles covering the events leading up to the opening and post-event coverage.

Contextual concepts. The articles' contextual concepts must be related to the 2021 Tokyo Olympics. Because of this, we decided the articles must first be related to the general concepts `Olympic Games` and `Japan`, which will limit the scope of retrieved articles. To further narrow the focus, we concentrate on seven different sports: `basketball`, `sports climbing`, `swimming`, `judo`, `rowing`, `skateboarding`, and `table tennis`. The sports were chosen for their diversity in competition organization and duration and their historic presence in previous Olympic Games. Notably, `sports climbing` and `skateboarding` debuted in the 2021 Olympic Games.

News collection

We can now retrieve the relevant news articles with the defined article criteria. To do this, we use Event Registry [113], the system that collects news articles from thousands of publishers, clusters them into news events, and enriches news articles and event clusters by extracting the named entities mentioned. It also links the article's textual components to corresponding Wikipedia pages through a process called wikification [129], [130]. Due to the structure of Wikipedia, where different-language Wikipedia pages corresponding to the same concept are linked, the wikification process identifies the Wikipedia pages in both the article's language and in English, if available.

The Event Registry system has a dedicated API (`https://www.newsapi.ai/`), which enables the retrieval of news articles and event cluster metadata. The retrieval can be done through multiple endpoints, which accept various parameters, including the language of the article, the publication date range, and the Wikipedia concepts the articles must relate to. Although the system provides clustered news articles, we used the API to collect only the news articles, as the clusters created by the system do not have the structure we are targeting.

Using the defined criteria, we developed code to retrieve the relevant articles automatically via the Event Registry API. We split the time range into days, then retrieved the articles published on a given day that were written in one of the selected languages and contained both the `Olympic Games` and `Japan` Wikipedia concepts, as well as at least one of the sports concepts listed in the criteria definition. The API allows using English Wikipedia concepts to retrieve articles in other languages, eliminating the need for prior translation.

Data set cleanup and preparation

The process retrieved 36,227 articles that matched the defined retrieval criteria. Each article contains various attributes, including the article's unique ID, its title and body, the

publication datetime, its language, its published URL address and the source publisher. We retrieved only the specifically mentioned attributes for each article, which are then used in the automatic clustering and manual annotation steps. We split the articles into seven data sets, each corresponding to one of the sports concepts. Table 5.2 shows the retrieved data set statistics.

Table 5.2: The data set statistics of the retrieved articles. Each language shows the number of retrieved articles, the percentage of the total data set, and the distribution of articles across the sports concepts. The least news articles were retrieved for Sport climbing.

Language	Overall (per cent of total)	Basketball	Sport climbing	Swimming	Judo	Rowing	Skateboarding	Table tennis
All	36,227	10,180	1,121	1,417	10,190	5,012	6,526	1,781
English	13,415 (37%)	3,980	883	176	2,474	2,236	3,331	335
Portuguese	6,742 (18%)	1,020	0	527	2,159	399	2,164	473
Spanish	6,261 (17%)	2,281	0	557	1,697	1,046	680	0
French	3,288 (9%)	1,054	4	118	1,574	249	0	289
Russian	2,677 (7%)	753	21	11	988	342	335	227
German	2,151 (6%)	551	125	10	655	447	0	363
Slovene	732 (3%)	390	87	0	135	18	10	92
Arabic	902 (3%)	140	0	15	483	258	6	0
Chinese	50 (0%)	11	1	1	20	17	0	0

5.1.2 News annotation

This section describes the steps used to annotate the news articles into clusters. The steps include automatic clustering of news articles and manual annotation and evaluation of clusters.

Automatic news clustering

We apply an online multilingual news clustering algorithm [239] to automatically cluster each dataset corresponding to one of the sport's concepts. This single-pass clustering algorithm processes the collected datasets by representing each article using its content embedding, the set of extracted named entities, and its publication datetime. The content embedding is created using the SBERT [84], [240] language model, trained to generate contextual embeddings appropriate for pairwise sentence similarity, while the named entities were extracted using the WikiNEuRal [241] multilingual named entity extraction model.

The representations for event clusters are created as aggregates of content embeddings, entity sets, and publication datetime values of articles within the clusters. The algorithm measures the content similarity, ratio of overlapping entities, and temporal proximity between the article and all clusters, checking if all three values are above the set thresholds defined at the beginning of the clustering process. The article is placed into the best event cluster that meets these criteria. If no such cluster exists, a new one is created containing the current article. The algorithm's hyper-parameters (thresholds) were chosen to optimize precision while maintaining a reasonable recall score. This intentional selection results in more clusters containing only the most similar articles, potentially reducing the inclusion of articles that should not be clustered together.

Post-automatic clustering data set statistics. The automatic news clustering algorithm created 16,049 unique clusters across the seven sports concepts. These clusters consist of news articles exhibiting similarity in content embeddings, shared named entities, and

temporal proximity as determined by their publication datetime. Table 5.3 shows the data set statistics after the automatic news clustering process.

Table 5.3: The data set statistics after the automatic news clustering process. For each language, it reports the number of articles (ART) and the number of clusters (CLS) generated. A cluster can contain articles from multiple languages, thus the sum of all clusters across the languages can be greater than the overall number of clusters.

	Overall		Basketball		Sport climbing		Swimming		Judo		Rowing		Skateboarding		Table tennis	
Language	ART	CLS	ART	CLS	ART	CLS	ART	CLS	ART	CLS	ART	CLS	ART	CLS	ART	CLS
All	36,227	16,049	10,180	4,103	1,121	566	1,417	772	10,190	3,575	5,012	2,209	6,526	2,346	1,781	848
English	13,415	6,270	3,980	1,935	883	452	176	124	2,474	1,148	2,236	1,067	3,331	1,385	335	159
Portuguese	6,742	2,794	1,020	540	0	0	527	296	2,159	755	399	202	2,164	809	473	192
Spanish	6,261	3,043	2,281	1,099	0	0	557	320	1,697	780	1,046	525	680	319	0	0
French	3,288	1,719	1,054	528	4	4	118	84	1,574	721	249	160	0	0	289	222
Russian	2,677	1,264	753	358	21	20	11	10	988	410	342	154	335	179	227	133
German	2,151	1,039	551	250	125	88	10	5	655	289	447	217	0	0	363	190
Slovene	732	366	390	189	87	40	0	0	135	71	18	11	10	8	92	47
Arabic	902	513	140	91	0	0	15	10	483	256	258	153	6	3	0	0
Chinese	50	41	11	11	1	1	1	1	20	16	17	12	0	0	0	0

Manual evaluation and annotation

Following the automatic news clustering process, the data sets underwent a manual evaluation and annotation. Each data set related to a sports concept was assessed using the following procedure, followed by their merger and final annotation.

Cluster representation. We have created a table for each cluster displaying its assigned articles and their values. Each table row contains the current cluster ID, the unique article ID, the article publication datetime, the article language, and the title and body. The article's URL and source were omitted from the table. Articles annotated in previous data sets were excluded from the current data set under consideration. Using this representation, the annotator has an overview of all the articles assigned to the cluster. Furthermore, to compare multiple clusters, the annotator can focus only on pairs of clusters and decide what action to take (more in *manual evaluation process*).

Manual annotation criteria. The annotation criteria is designed to categorize articles based on their responses to fundamental journalistic questions: who, what, where, when, and how. The first four questions yield objective answers, while the response to how can involve subjectivity. Because of this, alignment with the how question is considered an optional criterion. The annotators are tasks to group articles based on the responses to these questions. In addition, articles reflecting the same event from different perspectives (e.g. two articles describing the finals of a sport event, where one focuses on the gold and the other on the silver medalists) should be in the same event cluster, although they might not have the same responses to the above questions.

- Articles should predominantly focus on a singular event, situation, or key information.
- Articles must avoid resembling "click-bait," meaning they should not include phrases like "watch now" or "click here to see more." If they do, the article should be excluded from consideration.
- Articles should not primarily focus on presenting a schedule of events occurring on a specific day. If they do, the article should be excluded from consideration.

The intentionally ambiguous criterion aims to create diverse clusters of articles in various languages. Additionally, the criterion allows validation of an article based solely on its title, provided that the title is informative enough to provide the correct annotation.

Manual evaluation process. Given the evaluation criteria, the annotators reviewed the clusters and annotated the articles. If an annotator encountered a text that was not comprehensible, they were permitted to utilize machine translation services such as Google Translate (`https://translate.google.com/`). Annotators could remove specific articles from the dataset, relocate articles between clusters, merge, divide, and eliminate entire clusters. Each action taken was recorded and contributed to creating the final dataset. Afterward, the annotated datasets were merged and underwent a final evaluation, focusing on joining clusters reporting on the same event in different datasets and removing any inappropriate articles that might have been overlooked during the initial annotation pass.

Post-manual annotation data set statistics. Table 5.4 shows the data set statistics following the manual evaluation and annotation. The data shows a 70% reduction in the number of articles due to removing irrelevant news articles and event clusters corresponding to the schedules and "click-bait" like content; the number of articles was reduced by around 26k articles. This shows how much content of that nature is generated in a short time span. Furthermore, the number of clusters was also significantly reduced due to the removal and merging of event clusters. The final dataset was then formatted and prepared for analysis and publication.

Table 5.4: The data set statistics after the manual evaluation and annotation. For each language, it reports the number of articles (ART) and the number of clusters (CLS) generated. A cluster can contain articles from multiple languages, thus the sum of the number of clusters across the languages can be greater than the overall number of clusters.

	Overall		Basketball		Sport climbing		Swimming		Judo		Rowing		Skateboarding		Table tennis	
Language	ART	CLS	ART	CLS	ART	CLS	ART	CLS	ART	CLS	ART	CLS	ART	CLS	ART	CLS
All	10,940	1,350	2,898	368	101	21	275	52	3,868	487	1,217	197	2,203	219	378	50
English	4,009	729	963	189	63	15	31	7	1,096	234	670	138	1,048	159	138	15
Portuguese	2,410	368	293	83	0	0	106	25	981	150	108	21	822	88	100	16
Spanish	2,049	381	774	148	0	0	129	27	634	138	249	40	263	40	0	0
French	845	170	279	67	0	0	9	5	498	81	36	14	0	0	23	9
Russian	553	152	168	60	0	0	0	0	247	60	54	16	65	15	19	8
German	516	100	164	30	11	5	0	0	202	40	66	16	0	0	73	12
Slovene	331	102	216	73	27	5	0	0	56	13	3	1	4	3	25	9
Arabic	218	71	39	18	0	0	0	0	150	45	28	10	1	1	0	0
Chinese	9	5	2	2	0	0	0	0	4	2	3	2	0	0	0	0

5.1.3 The 2021 Tokyo Olympics data set

This section overviews the statistics for the final 2021 Tokyo Olympics data set, abbreviated as OG2021. It includes details and statistics about the annotated articles and clusters as well as language statistics demonstrating the data set's multilingual aspects.

Article statistics

Table 5.5 shows the statistics for the final data set, indicating a total of 10,940 articles organized into 1,350 clusters. The table also shows the average number of words in the article's title and body, the average cluster size, and their standard deviation for each language in the data set separately.

Table 5.5: The OG2021 data set statistics. It shows the number of articles, the average number of words in the title and body, the number of clusters, and the average cluster size. For Chinese, we report the average number of characters in the title and body.

Language	Script	Language family	Number of articles (per cent of total)	Average number of words in title (std)	Average number of words in body (std)	Number of clusters	Average cluster size (std)
All	-	-	10,940	-	-	1,350	8 (20)
English	Latin	Germanic	4,009 (37%)	11 (3)	1,231 (1,147)	729	5 (11)
Portuguese	Latin	Italic	2,410 (22%)	13 (3)	527 (374)	368	7 (12)
Spanish	Latin	Italic	2,049 (19%)	13 (4)	562 (427)	381	5 (9)
French	Latin	Italic	845 (8%)	13 (4)	565 (464)	170	5 (8)
Russian	Cyrillic	Slavic	553 (5%)	10 (3)	301 (358)	152	4 (6)
German	Latin	Germanic	516 (4%)	9 (3)	833 (1,011)	100	5 (6)
Slovenian	Latin	Slavic	331 (3%)	9 (3)	450 (370)	102	3 (3)
Arabic	Arabic	Semitic	218 (2%)	10 (3)	405 (269)	71	3 (3)
Chinese*	Chinese	Sinitic	9 (0%)	28 (7)	3,402 (1,550)	5	2 (1)

Article distribution over time. Figure 5.2 illustrates the distribution of articles over time. The articles were published between July 1, 2021 and August 14, 2021. The peak concentration of articles occurs between July 22, 2021, and August 8, 2021, corresponding to the opening and closing ceremonies of the Olympics. Furthermore, it also shows the distribution of articles for each language separately.

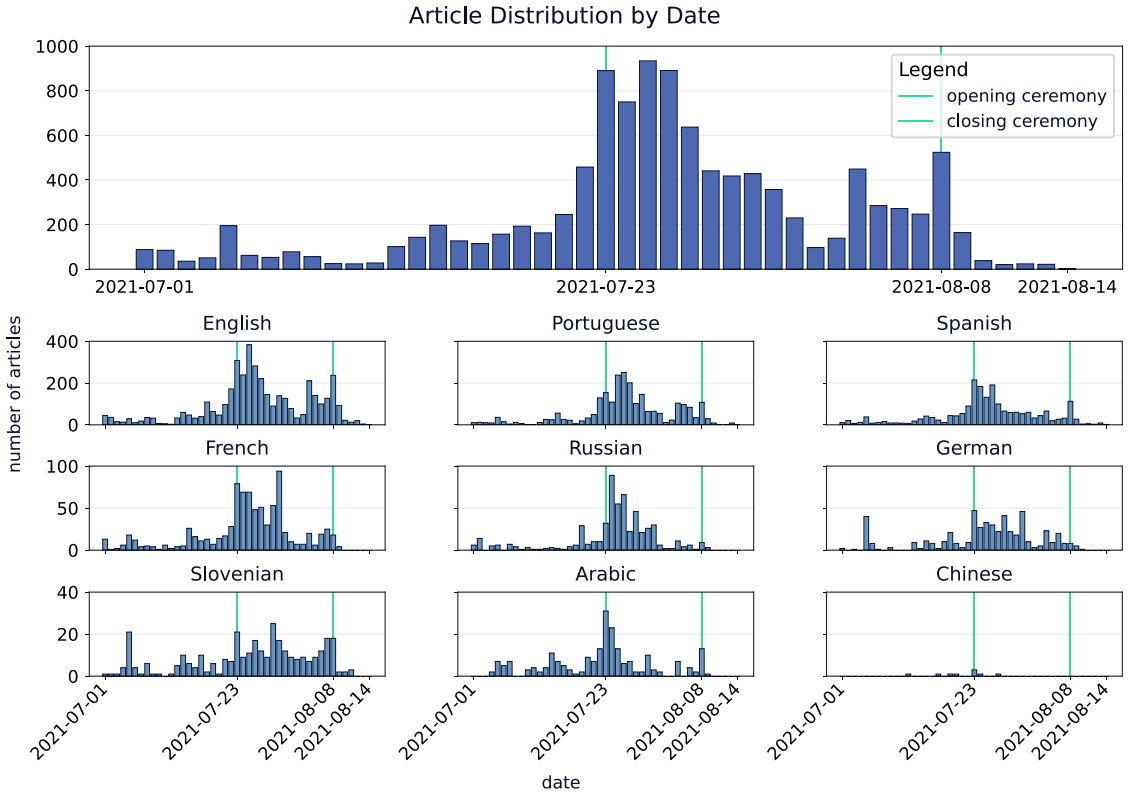


Figure 5.2: The OG2021 article distribution by date. Most articles were published between the Olympic Games' opening and closing ceremonies. The language-specific graphs are grouped in three rows representing the high, medium, and low present languages.

Cluster statistics

Cluster distribution based on their size. The data set comprises 1,350 distinct clusters, each containing approximately eight articles on average. Figure 5.3 shows the distribution of clusters based on their size, i.e., the number of assigned articles. Overall, around 95% of clusters contain 25 articles or fewer. Most clusters contain only two articles each, while the largest cluster includes 499 of them. This largest cluster pertains to the 2021 Tokyo Olympics opening ceremony and features coverage in multiple languages.

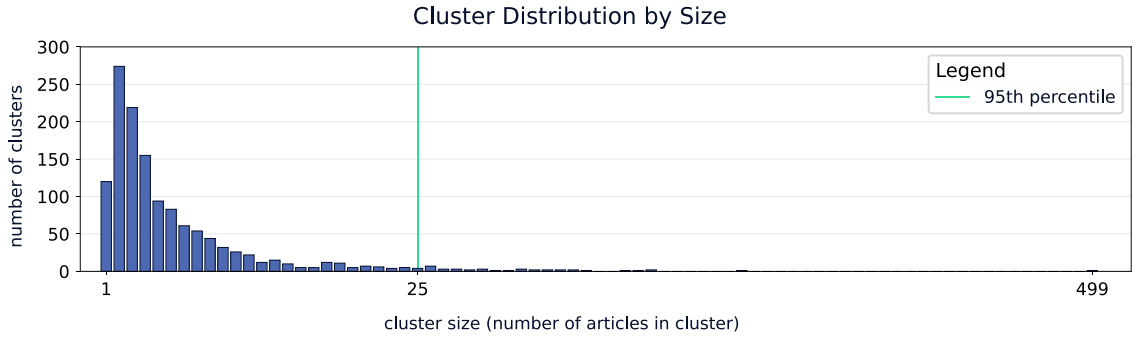


Figure 5.3: The OG2021 article distribution by size. About 95% of clusters contain 25 or fewer articles. The largest cluster contains almost 500 articles.

Cluster distribution per language. Figure 5.4 presents the distribution of clusters by languages. Most clusters contain articles written in just one language, with the most monolingual clusters corresponding to English, followed by Portuguese and Spanish. However, approximately 28% of the clusters consist of articles written in two or more languages. The most linguistically diverse cluster includes articles in nine languages, containing all the languages of the data set.

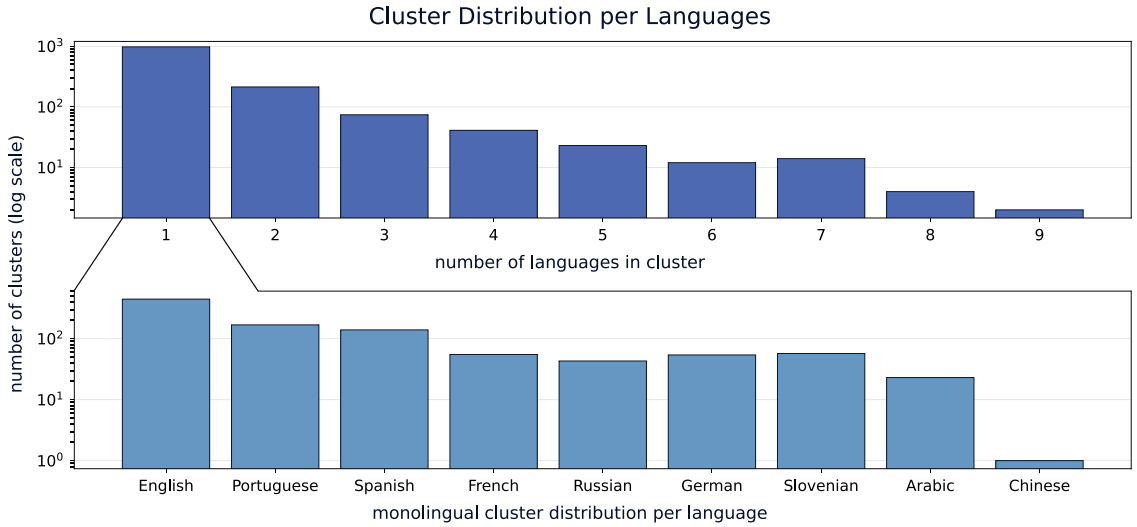


Figure 5.4: The OG2021 cluster distribution by language in log scale. The upper chart shows how many clusters contain a set number of languages; almost 28% of the clusters contain two or more languages. The bottom chart shows the distribution of monolingual clusters across languages.

Language co-occurrence in clusters. To highlight the multilingual nature of the data set, we computed the language co-occurrence across all clusters, as depicted in Figure 5.5. The diagonal of the co-occurrence matrix shows the percentage of clusters that include each language. The analysis reveals that any two languages appear together in at least one cluster, which represents 0.1% of all clusters. English, Portuguese, and Spanish language pairs co-occur in approximately 10% of the clusters. Other language pairs show co-occurrence rates between 1-6%. The only exception is Chinese, which has a minimal presence in the data set and appears in only nine articles.

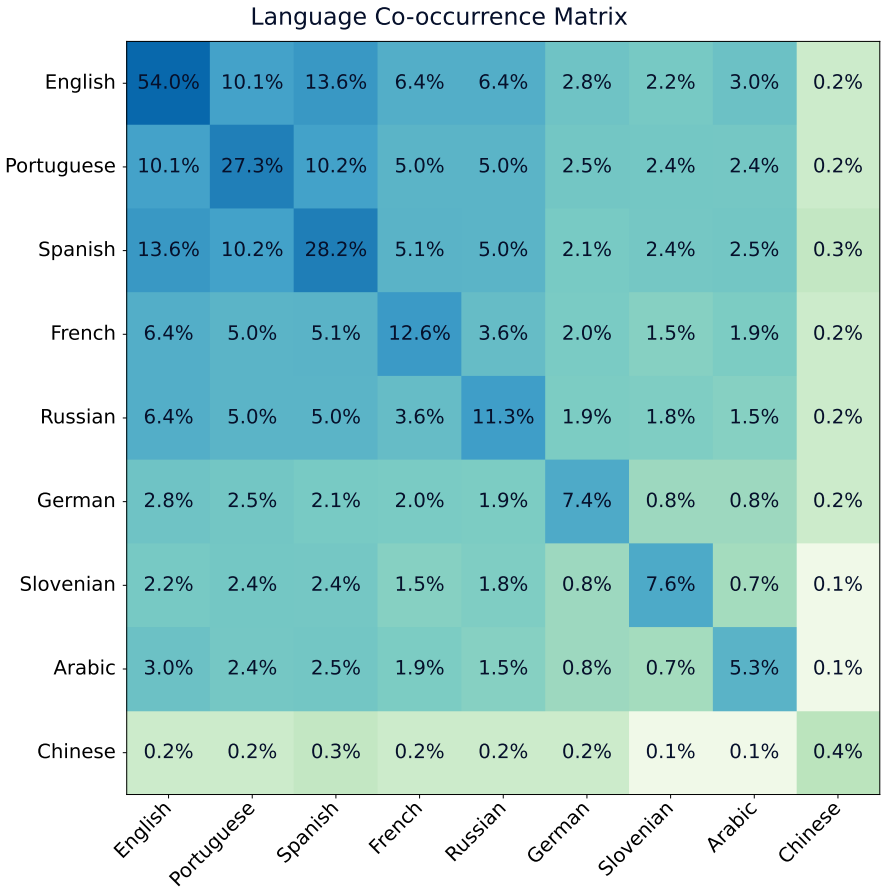


Figure 5.5: The OG2021 language co-occurrence in clusters. All language pairs co-occur in at least 0.1% of clusters.

The statistics demonstrate that the data set is diverse, covering a broad spectrum of languages and article clusters, particularly focused on the 2021 Tokyo Olympics period. The clusters vary in size, with most containing a modest number of articles. The multilingual analysis further highlights the data set's extensive language diversity, with each language appearing in at least 0.1% of all clusters.

5.1.4 Data set comparison

This section compares the OG2021 data set to the multilingual news article data set by Miranda et al. [114], referred to as CDET. Originally sourced from the Event Registry, this data set was developed to assess a cross-lingual news similarity and event tracking method [136]. It contains three primary languages, English, Spanish, and German, and six less-presented languages: Chinese, Slovenian, Croatian, French, Russian, and Italian.

Each article in the data set is characterized by its title and body, the language it is written in, the publication datetime, and its cluster label. The data set statistics are detailed in Table 5.6. It is frequently used to evaluate multilingual news clustering algorithms, making it a suitable benchmark for comparison.

Table 5.6: The CDET data set statistics. The statistics show the number of articles, the average number of words in the title and body, the number of clusters, and the average cluster size. For Chinese, we report the average number of characters in the title and body.

Language	Number of articles (per cent of total)	Average number of words in title (std)	Average number of words in body (std)	Number of clusters	Average cluster size (std)
Train set					
All	20,813	-	-	1,109	-
English	12,233 (59%)	9 (3)	436 (380)	593	21 (32)
Spanish	4,527 (22%)	11 (4)	355 (242)	416	11 (9)
German	4,043 (19%)	7 (3)	284 (220)	377	11 (7)
Chinese*	10 (0%)	20 (9)	446 (200)	1	10 (0)
Test set					
All	13,874	-	-	394	-
English	8,726 (63%)	9 (3)	537 (518)	222	39 (89)
Spanish	2,177 (16%)	11 (3)	401 (358)	149	15 (21)
German	2,101 (15%)	8 (3)	450 (496)	118	18 (45)
Chinese*	440 (3%)	19 (6)	1,445 (1,111)	9	49 (74)
Slovenian	37 (0%)	8 (3)	364 (232)	3	12 (3)
Croatian	13 (0%)	10 (4)	294 (301)	2	6 (0)
French	61 (0%)	10 (3)	338 (208)	2	30 (20)
Russian	231 (2%)	10 (5)	288 (294)	1	231 (0)
Italian	88 (0%)	9 (5)	506 (313)	2	44 (39)

Size and language comparison

We first compare the OG2021 and CDET data sets based on their size and article distribution per language. Table 5.7 shows the relevant statistics of the data sets side by side. First, the comparison shows that the CDET contains three times as many articles as OG2021. English articles are predominant in both data sets, comprising 60% of CDET and 37% of OG2021. Besides English, the CDET data set has a substantial number of Spanish and German articles, whereas OG2021 contains a significant amount of Spanish and Portuguese articles. The less-represented languages in OG2021 contain a higher percentage of articles, ranging from 0-6%, compared to those in the CDET data set, which range from 0-2%. Additionally, the CDET data set features larger average cluster sizes than those in OG2021; a cluster in the CDET has an average of 23 articles, whereas a cluster in OG2021 averages eight articles.

Cluster language comparison

Cluster distribution per language comparison. In this section, we analyze the cluster language distributions of the OG2021 and CDET data sets. Figure 5.6 shows the distribution of clusters by language within the CDET data set. It indicates that the vast majority of clusters consist of articles written in a single language, with a significant number of clusters also containing articles in two languages. Compared to OG2021, as shown in Figure 5.4, the CDET data set features fewer clusters with two or more languages (26%) than

Table 5.7: The OG2021 vs CDET data set comparison statistics. The statistics show the number of articles and clusters found in each data set. While OG2021 has less articles, they are more distributed across different languages than those in the CDET data set.

	OG2021		CDET	
Language	# articles	# clusters	# articles	# clusters
All	10,940	1,350	34,687	1,503
English	4,009 (37%)	729	20,959 (60%)	815
Spanish	2,046 (19%)	381	6,704 (19%)	565
German	516 (4%)	100	6,144 (18%)	495
Chinese	9 (0%)	5	450 (2%)	10
Slovenian	331 (3%)	102	37 (0%)	3
Croatian	-	-	13 (0%)	2
French	845 (8%)	170	61 (0%)	2
Russian	553 (5%)	152	231 (1%)	1
Italian	-	-	88 (0%)	2
Portuguese	2,410 (22%)	368	-	-
Arabic	218 (2%)	71	-	-

OG2021 (28%). Additionally, OG2021 includes a higher number of clusters containing between three and nine languages and has clusters that include as many as eight and nine languages, which are not present in the CDET data set.

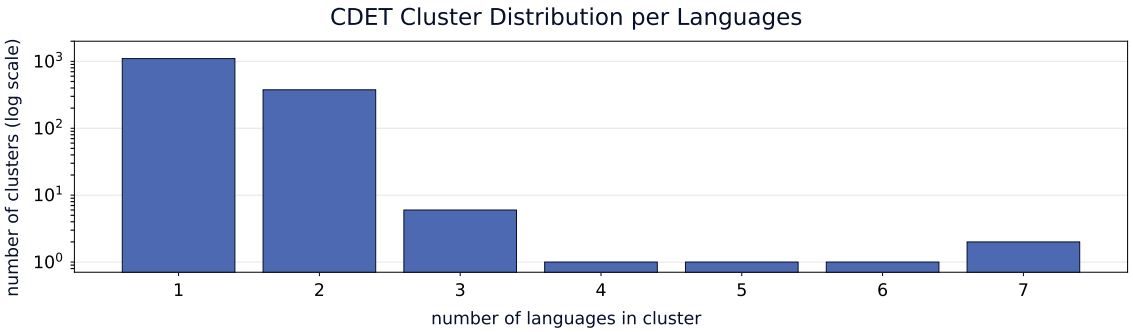
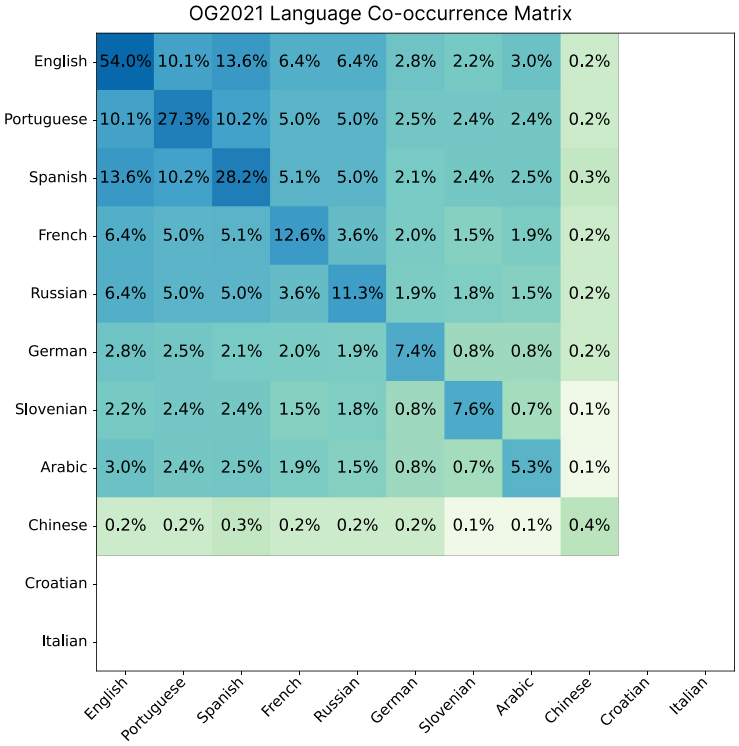


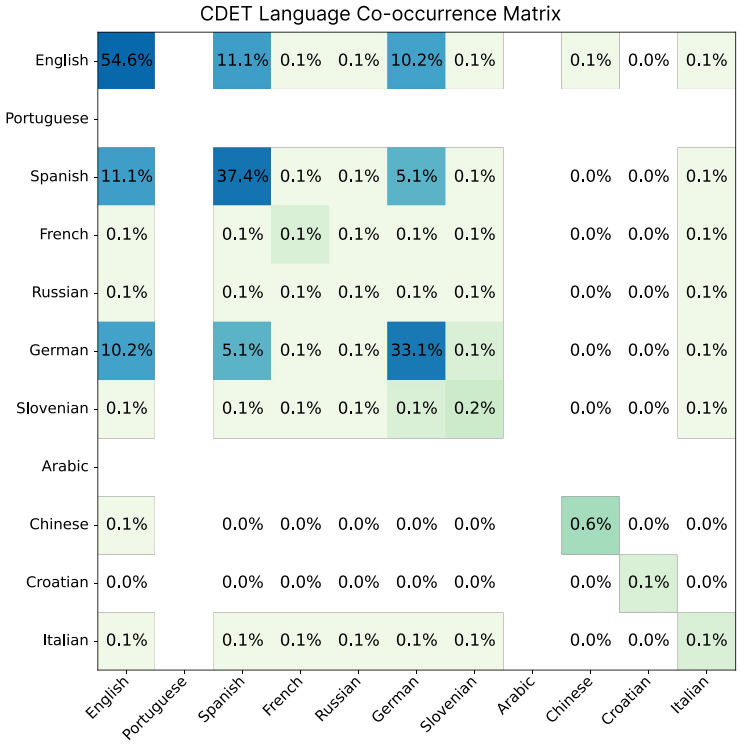
Figure 5.6: The CDET cluster distribution by language. About 26% of the clusters contain two or more languages.

Language co-occurrence comparison. To evaluate the multilingual nature of the data sets, we computed the language co-occurrence across the CDET clusters and compared it to that of OG2021, as depicted in Figure 5.7. The OG2021 data set exhibits a higher rate of language co-occurrence across its clusters than CDET. In the CDET data set, only English, Spanish, and German articles have high co-occurrence, which reflects its article language distribution. Additionally, some language pairs in CDET do not co-occur at all, whereas, in OG2021, every language pair appears in at least one cluster.

The comparison of the two data sets shows that while they have some attributes in common, such as the present languages and the average number of words in the title and body, they vary in their article language distribution and the language co-occurrence in clusters. However, both data sets are valuable and helpful for evaluating multilingual news clustering algorithms.



(a) The OG2021 language co-occurrence matrix.



(b) The CDET language co-occurrence matrix.

Figure 5.7: The OG2021 vs CDET language co-occurrence comparison. Missing values denote the absence of the language in the data set.

5.2 Online Multilingual News Clustering Using Wasserstein Distance

Online news sources generate hundreds of thousands of articles daily, covering significant events worldwide. These events span diverse domains, such as politics, sports, culture, and industry, and are presented in various languages. Furthermore, articles may provide different perspectives on the same event, influenced by political, social, or demographic biases. Collecting and grouping articles that report on the same event enables us to fully analyze it, aiding our understanding of its global importance and implications for the world.

Online news clustering poses a significant challenge due to the global dynamic nature and vast volume of news. News is published at a high frequency, which is why the articles usually need to be handled individually as they appear in the news stream. The system must then decide whether to include an article in a specific event cluster. The multilingual nature of articles adds complexity, requiring syntax-independent features to cluster the articles effectively. Furthermore, articles may report on events occurring at various timestamps in the past. While these events may be semantically similar, such as a volcano eruption in the same location, they happened at different times. Because of this, online news clustering systems must leverage both text and time-related features to appropriately group articles.

Online news clustering algorithms aggregate article information to determine how to group articles into distinct clusters. Most algorithms define a cluster as a collection of articles with similar characteristics aggregated within this definition. However, the process of determining if two clusters should be merged relies on a comparison of their aggregated values. This approach, while useful, does not provide a comprehensive understanding of the clusters, as a single article can significantly alter the representations of both clusters. Considering all of the article's information and representations, without aggregates, would improve the decision when to merge two clusters.

This section aims to develop an online multilingual news clustering algorithm that uses semantic technologies and unconventional similarity methods to compare news across different languages and incorporates a temporal component to assess the temporal relevance of assigning an article to a specific event cluster.

The main research objectives are to (1) develop the algorithm and evaluate its performance on a news clustering evaluation data set of various languages and (2) assess how unconventional similarity methods contribute to its performance.

We propose an online multilingual news clustering algorithm that operates in two stages. The first stage creates article clusters. In contrast, the second stage merges the generated article clusters into event clusters based on their Wasserstein distance and temporal similarity. The algorithm aims to limit the amount of information aggregation done during clustering, thereby incorporating as much available information about the articles as possible. The algorithm is evaluated on the CDET and OG2021 data sets, described in Chapter 5.1. The evaluation metrics used include the standard F1 score and the BCubed F1 score. The results show that the proposed algorithm performs on par with the compared algorithms that utilize classification models. Additionally, the analysis shows that the proposed cluster merging approach, leveraging Wasserstein distance, significantly enhances the quality of the generated clusters compared to not merging the clusters. It can thus serve as an alternative to classification models for cluster merging decisions.

The remaining sections are structured as follows. We first describe the online news clustering algorithm in Section 5.2.1 and its implementation details in Section 5.2.2. Next, we present the experiment setting in Section 5.2.3, describing the data set, evaluation metrics, and comparing algorithms. Finally, the experiment results are reported in Section 5.2.4.

5.2.1 Algorithm description

We propose a two-stage online multilingual news clustering algorithm, called *Wasserstein-based news Article Clustering* (WAC), illustrated in Figure 5.8. In the first stage, the algorithm focuses on clustering news articles. Each pre-existing cluster is ranked based on its content and temporal similarity to the current article. Based on the cluster's rank score, the article is assigned either to the highest-ranking cluster or a newly created one. In the second stage, the algorithm merges the article clusters to form the final event clusters. Each previously seen event cluster is first ranked based on different metrics for measuring the content and temporal similarity to the current article cluster. Again, given the rank score, we merge the article cluster into either the highest-ranking event cluster or a newly created one.

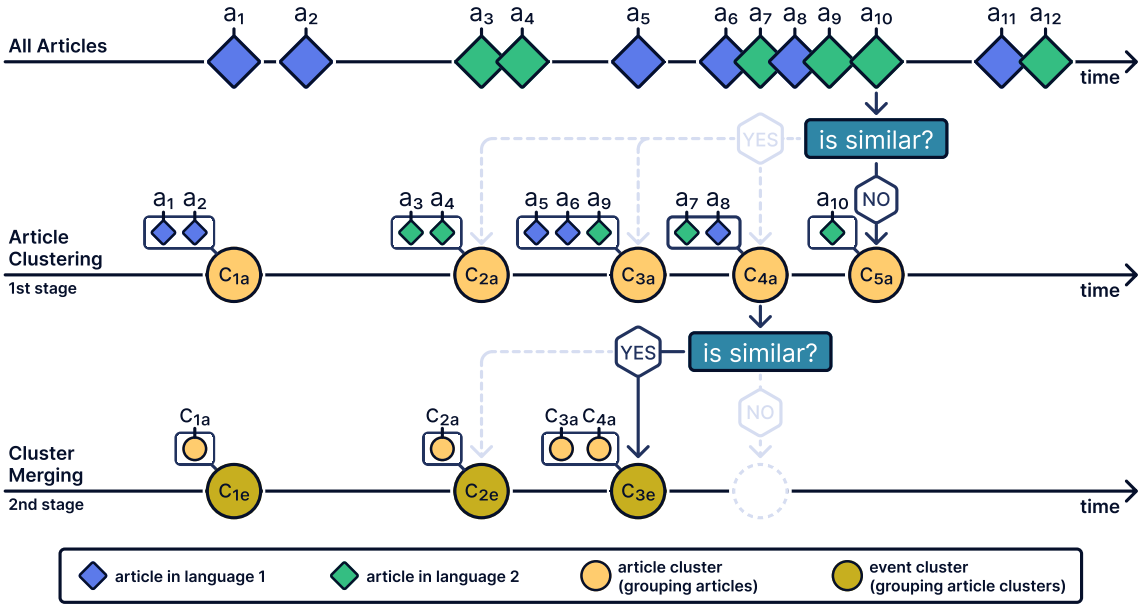


Figure 5.8: The WAC algorithm diagram. The time-ordered articles are first clustered into article clusters based on the clustering similarity ranking, resulting in the clusters depicted in the first stage. Next, the article clusters are clustered into event clusters based on the Wasserstein distance and temporal similarity, resulting in event clusters shown in the second stage.

In the following sections, we provide details on how the algorithm works. We describe how articles are represented and explain the components used and the procedure of both article clustering and cluster merging.

Article representation

Every news article a is presumed to have a title and body, accompanied by its publication datetime and the language in which it is written. We first format the title and body by replacing multiple whitespaces with a single space. Using this information, we proceed to generate the article embeddings and extract named entities found in the article.

Article embeddings. Each article is assigned two embeddings representing the title and body, respectively. We do this to retain the information in both the title and body, as the body can contain more information, possibly unrelated to the title. With this, we also want

to find articles that contain similar information in both the title and the body. Using a multilingual language model trained for generating vectors for cross-lingual clustering tasks, we create the corresponding embeddings by separately inputting the article's title and body into the language model, resulting in the vectors that capture the semantic meaning of its input. We denote the title and body embeddings as v_a^{title} and v_a^{body}, respectively.

Article named entities. We extract the relevant entities associated with every article. Using a multilingual named entity recognition model, we identify the named entities mentioned in the article's title or body and join them in a standard set. The entities extracted are in the same language as the article. This enables an additional way of comparing articles within the same language. Afterward, we eliminate duplicates and retain the remaining unique entities for future use. We denote the set of unique entities in the article as e_a.

Final article representation. Given the above definitions, we represent each article a as a set of the following attributes: the article title embedding $\vec{v}_a^{title}$, the article body embedding $\vec{v}_a^{body}$, the article named entities e_a, the article publication time t_a, and the article language $a.\mathsf{lang}$. For the remainder of the chapter, we use the following notation $a = \{\vec{v}_a^{title}, \vec{v}_a^{body}, e_a, t_a, a.\mathsf{lang}\}$.

Article clustering phase

The goal of article clustering is to group articles such that the articles in the group depict the same story. Our approach builds upon previous work [239]. To ensure clarity and completeness, we describe the original methodology and the adjustments made.

Cluster representations. Each cluster is characterized by representations derived from its articles. The following describes these representations and how they are created.

Cluster centroid embeddings. The centroid vectors represent the central theme of the cluster articles. It is primarily used to assess if an incoming article is similar enough to the articles in the cluster. The centroid is calculated as the average attribute-specific embedding of all articles within that cluster:

$$\vec{v}_c^{attr} = \frac{1}{n} \sum_{i=1}^{n} \vec{v}_{a_i}^{attr},$$

where n is the number of articles in the cluster, and $\vec{v}_{a_i}^{attr}$ represents the article's attribute embedding, $attr \in \{title, body\}$.

Cluster named entities. Each cluster has a set of named entities and is used to assess if the incoming article mentions the event's entities. The set is a union of named entities extracted from the articles within the cluster:

$$e_c = \bigcup_{i=1}^{n} e_{a_i},$$

where e_{a_i} represents each article's named entities.

Cluster temporal aggregates. The cluster's temporal attributes are used to assess if the incoming article was published at a time when it could still report about the cluster-represented event. These are represented as a tuple of the minimum, average and maximum article datetime value, depicted as

$$t_c = \left(t_c^{\min}, t_c^{\text{avg}}, t_c^{\max}\right),$$

where $t_c^{\texttt{aggr}}$ is defined as the aggregate of the publication datetime value of the articles contained in the cluster c, i.e. $t_c^{\texttt{aggr}} = \texttt{aggr}\left(\{t_{a_i}\}_{i=1}^{n}\right)$ where t_{a_i} is the publication datetime of article $a_i \in c$ and $\texttt{aggr} \in \{\min, \text{avg}, \max\}$ is the aggregate function.

Final cluster representation. Given the above definitions, we define each cluster c as a set of the following attributes: the cluster title centroid embedding $\vec{v}_c^{title}$, the cluster body centroid embedding $\vec{v}_c^{body}$, the cluster named entities e_c, and the cluster temporal aggregates t_c. For the remainder of the chapter, we use the following notation $c = \{\vec{v}_c^{title}, \vec{v}_C^{body}, e_c, t_c\}$.

The similarity score methods. The article clustering is performed using three main methods for measuring the temporal, content, and entity similarity between the article and cluster, respectively. What follows are the implementation details of each of these methods.

Temporal score. The temporal score between two datetime values is measured using Equation 5.1. It takes two datetime values t_1 and t_2 and the scaling factor σ denoting the rate at which the temporal score diminishes towards zero if the respective datetime values are not close. First, it measures the absolute distance between the two datetime values, $\Delta t = |t_1 - t_2|$. Afterwards, it uses Δt to calculate the exponent $z = -\frac{1}{2}\left(\frac{\Delta t}{\sigma}\right)^2$. The final temporal score is computed as the exponential value e^z. The equation returns higher values when datetime values t_1 and t_2 are closer in proximity.

$$\texttt{TemporalScore}(t_1, t_2, \sigma) = e^{-\frac{1}{2}\left(\frac{|t_1 - t_2|}{\sigma}\right)^2} \tag{5.1}$$

The article-cluster temporal score. The temporal score between article a and cluster c is calculated using Algorithm 5.1. The score shows how temporally aligned the article is to the articles in the cluster. The algorithm calculates the average temporal score between the article's publication datetime value t_a and each of the cluster's temporal attributes in t_c using the Equation 5.1 and the predetermined scaling factor σ. Higher values indicate better temporal matching between article a and cluster c.

Algorithm 5.1: The article-cluster temporal score: `ACTemporalScore`

Input : The article a; the cluster c; the scaling factor σ
Output: The article-cluster temporal score

```
tscores ← list()                                          // empty list
for aggr ∈ {min, avg, max} do
    score ← TemporalScore(t_a, t_c^aggr, σ)               // see Equation 5.1
    tscores.append(score)
end
return avg(tscores)
```

The article-cluster content score. The article-cluster content score depicts the content alignment between article a and cluster c. It is calculated by determining the pairwise cosine similarity between the title $(\vec{v}_a^{title}, \vec{v}_c^{title})$ and body $(\vec{v}_a^{body}, \vec{v}_c^{body})$ embeddings of the article and cluster. Additionally, the cosine similarity between the title and body embeddings is measured to determine how similar the article's title is to the cluster's aggregated body. Similar is done for the cluster's aggregated title and the article's body. The final score is the average of all computed similarity scores. Higher values indicate better content matching between article a and cluster c. The method is depicted in Algorithm 5.2.

Algorithm 5.2: The article-cluster content score: `ACContentScore`

Input : The article a; the cluster c
Output: The article-cluster content score

```
cscores ← list()                                          // empty list
for attr_1 ∈ {title, body} do
    for attr_2 ∈ {title, body} do
        score ← CosineSimilarity(v_a^attr_1, v_c^attr_2)   // see Equation 2.2
        cscores.append(score)
    end
end
return avg(cscores)
```

The article-cluster entity score. This score measures the named entities' overlap between article a and cluster c. If either of the entity sets e_a or e_c are empty, then the score is set to zero. Otherwise, we calculate the ratio between the intersection of the entity sets and each entity set, respectively, showing how many entities in the article set are also present in the cluster set and vice-versa. We return the average of the calculated ratios as the final score, where higher values indicate better entity overlap. The method is depicted in Algorithm 5.3.

Algorithm 5.3: The article-cluster entity score: `ACEntityScore`

Input : The article a; the cluster c
Output: The article-cluster entity score

if e_a *is empty* **or** e_c *is empty* **then**
    return 0
end
escores ← list($\frac{|e_a \cap e_c|}{|e_a|}$, $\frac{|e_a \cap e_c|}{|e_c|}$)
return avg(escores)

Article clustering algorithm. The online article clustering is described in Algorithm 5.4. The inputs are the current article a, the given active cluster set C, and the hyper-parameters which include the rank threshold $\tau_{\text{RANK}}^{\text{CLUSTER}}$, the named entity threshold $\tau_{\text{ENTS}}^{\text{CLUSTER}}$, and the temporal scaling factor σ_{CLUSTER}. The algorithm returns the updated cluster set C, with the article a assigned to an existing or a newly created cluster.

The algorithm first prepares the relevant cluster set denoted as C_{REL}. If performing monolingual article clustering, the appropriate cluster set is comprised of clusters $c_i \in C$, which contain only articles in the same language as the current article a, i.e. $a.\mathsf{lang} = c_i.\mathsf{lang}$, where $c_i.\mathsf{lang}$ depicts the language of the articles in the cluster. Otherwise, we set the relevant set containing all active clusters, $C_{\text{REL}} = C$. If the set of relevant clusters is empty, a new cluster containing the current article a is created and added to the cluster set C. Otherwise, for each relevant cluster, we calculate their rank by multiplying their temporal score (see Algorithm 5.1) and content score (see Algorithm 5.2). Afterwards, we sort the clusters based on their rank in descending order and consider only those with the rank above the threshold $\tau_{\text{RANK}}^{\text{CLUSTER}}$. We then iterate through the clusters $c_i \in C_{\text{REL}}$ and search for the first one which has the entity score above the set entity threshold $\tau_{\text{ENTS}}^{\text{CLUSTER}}$. Once we find it, we assign the article a to the cluster and update the active set C. If no such cluster exists, we create a new cluster containing the target article and add it to the active set C.

Algorithm 5.4: The article clustering algorithm.

Input : The article a; the set of active clusters $C = \{c_i\}_{i=1}^{k}$; the rank $\tau_{\text{RANK}}^{\text{CLUSTER}}$ and named entity $\tau_{\text{ENTS}}^{\text{CLUSTER}}$ thresholds; the temporal scaling factor σ_{CLUSTER}
Output: The updated set of clusters C

$C_{\text{REL}} \leftarrow C$ `// copy the active clusters`
if $|C_{\text{REL}}| = 0$ **then**
  create a new cluster containing the article $c_{k+1} \leftarrow a$
  update the active set $C \leftarrow c_{k+1}$
  return C `// proceed with article clustering cleanup`
end
foreach $c_i \in C_{\text{REL}}$ **do**
  $c_i.\text{rank} \leftarrow$ `ACTemporalScore`$(a, c_i, \sigma_{\text{CLUSTER}})$ `// see Algorithm 5.1`
    $\times$ `ACContentScore`(a, c_i) `// see Algorithm 5.2`
end
$C_{\text{REL}} \leftarrow$ `sort`$(C_{\text{REL}}, c_i.\text{rank}, \text{descending})$
foreach $c_i \in C_{\text{REL}}$ **where** $c_i.\text{rank} \geq \tau_{\text{RANK}}^{\text{CLUSTER}}$ **do**
  escore $\leftarrow$ `ACEntityScore`(a, c_i) `// see Algorithm 5.3`
  if escore $\geq \tau_{\text{ENTS}}^{\text{CLUSTER}}$ **then**
    assign the article to the cluster $c_i \leftarrow a$
    update the active set $C \leftarrow c_i$
    return C `// proceed with article clustering cleanup`
  end
end
create a new cluster containing the article $c_{k+1} \leftarrow a$
update the active set $C \leftarrow c_{k+1}$
return C `// proceed with article clustering cleanup`

Article clustering cleanup. After assigning article a to a cluster, active set C is sent through a cleanup process described in Algorithm 5.5. The cleanup aims to remove clusters significantly older than the processed article a, thereby reducing the number of clusters the article clustering algorithm must handle. For each cluster $c_i \in C$, the algorithm calculates the temporal score given article a and temporal scaling factor σ_{CLUSTER}. If the score is below or equal to a set threshold $\tau_{\text{TIME}}^{\text{CLUSTER}}$, the associated cluster is removed from the active cluster set. After evaluating and removing all clusters as necessary, the clustering algorithm proceeds by processing the following news article.

Algorithm 5.5: The article clustering cleanup.

Input : The current article a; the set of active clusters $C = \{c_i\}_{i=1}^{k}$; the temporal threshold $\tau_{\text{TIME}}^{\text{CLUSTER}}$; the temporal scaling factor σ_{CLUSTER}
Output: The updated active clusters set C

foreach $c_i \in C$ **do**
  tscore $\leftarrow$ `ACTemporalScore`$(a, c_i, \sigma_{\text{CLUSTER}})$ `// see Algorithm 5.1`
  if tscore $\leq \tau_{\text{TIME}}^{\text{CLUSTER}}$ **then**
    remove the cluster c_i from the active cluster set C
  end
end
return C

Cluster merging phase

The article clustering algorithm may produce multiple clusters that depict the same event. To address this problem, we developed the cluster merging component to group article clusters representing the same story. The main component, the Wasserstein distance introduced in Appendix 3, measures the similarity between two clusters given their corresponding articles. In this section, we describe the methods for measuring temporal and content similarity between clusters, followed by describing the cluster merging algorithm.

The similarity score methods. The cluster merging is performed using two main methods, one for measuring the temporal similarity and another for measuring the content similarity. What follows is the description of both methods.

The cluster-cluster temporal similarity score. The temporal similarity score between clusters c_i and c_j, described in Algorithm 5.6, shows the temporal alignment of the two clusters. We first calculate the individual temporal scores between each of the temporal aggregates of cluster c_i and c_j given the temporal scaling factor σ, i.e. $\texttt{TemporalScore}(t_{c_i}^{aggr_1}, t_{c_j}^{aggr_2}, \sigma)$ where $aggr_1, aggr_2 \in \{\min, \text{avg}, \max\}$, and structure them into a matrix T. Taking inspiration from BERTScore [103], we then calculate the temporal F1 score, where the precision and recall are defined as the average maximum values of the corresponding rows and columns in matrix T, respectively. The F1 score is returned as the final temporal score between the two clusters.

Algorithm 5.6: The cluster-cluster temporal score: `CMTemporalScore`

Input : The clusters c_i and c_j; the scaling factor σ
Output: The cluster-cluster temporal score

```
aggrs ← list(min, avg, max)
n ← len(aggrs)
T ← matrix(n, n)                                   // matrix with all zeros
for k ← 1 to n do
    for ℓ ← 1 to n do
        T_{k,ℓ} ← TemporalScore(t_{c_i}^{aggrs[k]}, t_{c_j}^{aggrs[ℓ]}, σ)      // see Equation 5.1
    end
end
P = avg(max(T, dim = 1))                  // average maximum value by rows
R = avg(max(T, dim = 2))                  // average maximum value by columns
F1 = 2 P·R / (P+R)
return F1
```

The cluster-cluster content similarity score. The content similarity score between two clusters c_i and c_j is measured using the Wasserstein distance between the probability distributions of the cluster's corresponding articles (see Algorithm 5.7). Let $c_i = \{a_{ik}\}_{k=1}^{n}$ and $c_j = \{a_{j\ell}\}_{\ell=1}^{m}$ be two clusters containing their distinct articles. Let $D \in \mathbb{R}^{n \times m}$ be the matrix of distances between articles of both clusters. The distance used is the cosine distance between the article's embeddings:

$$D_{k,\ell} = d(a_{ik}, a_{j\ell}) = \frac{1}{|\Lambda|} \sum_{\lambda \in \Lambda} 1 - \frac{\langle \vec{v}_{a_{ik}}^{\lambda}, \vec{v}_{a_{j\ell}}^{\lambda} \rangle}{\|\vec{v}_{a_{ik}}^{\lambda}\|_2 \|\vec{v}_{a_{j\ell}}^{\lambda}\|_2},$$

where $\Lambda = \{title, body\}$ is the set of article's content attributes. We use multilingual language models to generate the articles' content embeddings, enabling us to compare

articles in different languages. However, the embeddings of texts in different languages are generally further apart. To compensate for this, we update matrix D such that the values corresponding to article pairs that are in different languages are reduced by a factor of $\eta \in (0,1)$ and thus closer in the embedding space:

$$D_{k,\ell} = \begin{cases} \kappa D_{k,\ell}, & a_{ik}.\mathsf{lang} \neq a_{j\ell}.\mathsf{lang}, \\ D_{k,\ell}, & \text{otherwise.} \end{cases}$$

Through experimentation, we found that $\kappa = 0.99$ gives the best performance. Let μ_{c_i} and ν_{c_j} be the article distributions of the two clusters:

$$\mu_{c_i} = \sum_{k=1}^{n} w_{a_{ik}} \delta_{a_{ik}}, \qquad \sum_{k=1}^{n} w_{a_{ik}} = 1,$$
$$\nu_{c_j} = \sum_{\ell=1}^{m} w_{a_{j\ell}} \delta_{a_{j\ell}}, \qquad \sum_{\ell=1}^{m} w_{a_{j\ell}} = 1,$$

where $\delta_{a_{ik}}$, $\delta_{a_{j\ell}}$ are the Dirac delta functions as defined with Equation (3.7). Weights $w_{a_{ik}}$ and $w_{a_{j\ell}}$ represent the mass of their respective articles a_{ik} and $a_{j\ell}$. Although we can assign weights to signify different aspects of the articles, we treat all articles equally important in the cluster. Therefore, we set the weights to be uniformly distributed, meaning $w_{a_{ik}} = \frac{1}{n}$, $k = 1, \ldots, n$ and $w_{a_{j\ell}} = \frac{1}{m}$, $\ell = 1, \ldots, m$. Using distance matrix D and article distributions μ_{c_i}, ν_{c_j}, we calculate the Wasserstein distance between the two clusters by solving the optimization problem depicted in Equation (4.3). The solution also returns the transportation matrix $P \in \mathbb{R}^{n \times m}$, which is used to calculate the final distance:

$$d_W(c_i, c_j) = \sum_{k=1}^{n} \sum_{\ell=1}^{m} P_{k,\ell} D_{k,\ell}.$$

Afterwards, we use the distance value to calculate the final value using the following equation:

$$\texttt{CMContentScore}(c_i, c_j) = e^{-d_W(c_i,c_j)^2} \in (0,1].$$

The final value signifies the content similarity score between the two clusters, where higher values indicate better content matching. The details on the Wasserstein distance hyperparameters are in Section 5.2.2.

The cluster merging algorithm. Algorithm 5.8 outlines the cluster merging process. It begins with the current article cluster c, the set of active event clusters $C = \{c_i\}_{i=1}^{k}$, and the hyper-parameter set containing the relevancy rank score threshold $\tau_{\text{RANK}}^{\text{MERGE}}$, and the temporal scaling factor σ_{MERGE}. It then outputs the updated set of event clusters C, where the current article cluster c is either merged with an existing one or added to the set independently.

First, we check if the cluster set C is empty. If so, cluster c is added to the active cluster set C. Otherwise, we create the cluster set $C_{\text{REL}} \subseteq C$ containing the event clusters most similar to c based on their temporal (see Algorithm 5.6) and centroid embedding similarity (an extension of Algorithm 5.2). Set C_{REL} is designed to reduce the number of event clusters compared to the article cluster c, thereby decreasing the time required to execute the algorithm. Afterwards, the algorithm finds the cluster $c_{\text{BEST}} \in C_{\text{REL}}$ that is most similar to c based on the product of its temporal and content scores, described above. If the product is greater than the rank score threshold $\tau_{\text{RANK}}^{\text{MERGE}}$, the articles of cluster c are assigned to cluster c_{BEST} and the active event cluster set C gets updated. Otherwise, cluster c is added to the event cluster set C.

Algorithm 5.7: The cluster-cluster content score: `CMContentScore`

Input : The clusters c_i and c_j; the Wasserstein-related parameters γ and nit
Output: The cluster-cluster content score

attrs ← `list`$(title, body)$
n, m ← `len`$(c_i$.articles$)$, `len`$(c_j$.articles$)$
D ← `matrix`(n, m) `// matrix with all zeros`
for $k \leftarrow 1$ **to** n **do**
  for $\ell \leftarrow 1$ **to** m **do**
    $a_{ik}, a_{j\ell} \leftarrow c_i$.articles$[k], c_j$.articles$[\ell]$
    foreach $attr \in$ attrs **do**
      $\mathsf{D}_{k,\ell} \leftarrow \mathsf{D}_{k,\ell}$ + `CosineDistance`$(v_{a_{ik}}^{attr}, v_{a_{j\ell}}^{attr})$ `// see Equation 2.2`
    end
    $\mathsf{D}_{k,l} \leftarrow \frac{\mathsf{D}_{k,\ell}}{\texttt{len(attrs)}} \times (\kappa$ **if** a_{ik}.lang $\neq a_{j\ell}$.lang **else** $1)$
  end
end
dist(c_i), dist(c_j) ← `list`$(\frac{1}{\mathsf{n}}$, size = n$)$, `list`$(\frac{1}{\mathsf{m}}$, size = m$)$
W ← `SinkhornKnopp`(D, dist(c_i), dist(c_j), γ, nit) `// see Algorithm 3.1`
return $\exp(-\mathsf{W}^2)$

Algorithm 5.8: The cluster merging algorithm.

Input : The current article cluster c; the set of active event clusters $C = \{c_i\}_{i=1}^{k}$; the rank threshold $\tau_{\text{RANK}}^{\text{MERGE}}$; temporal scaling factor σ_{MERGE}; the Wasserstein-related parameters γ and nit
Output: The updated set of active event clusters C

if *there are no clusters in* C **then**
  add the cluster c to the active cluster set C
  return C `// proceed with cluster merging cleanup`
end
Create the subset $C_{\text{REL}} \subseteq C$ containing clusters that are most similar to c in terms of temporal and centroid similarity
foreach $c_i \in C_{\text{REL}}$ **do**
  c_i.rank ← `CMTemporalScore`$(c, c_i, \sigma_{\text{MERGE}})$ `// see Algorithm 5.6`
    $\times$ `CMContentScore`(c, c_i) `// see Algorithm 5.7`
end
$c_{\text{BEST}} \leftarrow$ highest ranking cluster in C_{REL}
if c_{BEST}.rank $\geq \tau_{\text{RANK}}^{\text{MERGE}}$ **then**
  assign the articles to the event cluster $c_i \leftarrow c$.`articles`
  update the active cluster set $C \leftarrow c_i$
  return C `// proceed with cluster merging cleanup`
end
add the cluster c to the active cluster set C
return C `// proceed with cluster merging cleanup`

The cluster merging cleanup. After processing the current cluster c, the active event cluster set C is sent through a cleanup process described in Algorithm 5.9. For each cluster $c_i \in C$, we calculate the temporal score given the article cluster c and temporal scaling factor σ_{MERGE}. If the score is below or equal to a set threshold $\tau_{\text{TIME}}^{\text{MERGE}}$, the associated cluster is

removed from the active event cluster set, thus reducing the number of clusters that are considered in Algorithm 5.8. Afterwards, the cluster merging continues with the following article cluster.

Algorithm 5.9: The cluster merging cleanup.

Input : The current cluster c; the set of active clusters $C = \{c_i\}_{i=1}^{k}$; the temporal threshold $\tau_{\text{TIME}}^{\text{MERGE}}$; the temporal scaling factor σ_{MERGE}
Output: The updated set of active event clusters C

```
foreach c_i ∈ C do
    tscore ← CMTemporalScore(c, c_i, σ_MERGE)          // see Algorithm 5.6
    if tscore ≤ τ_TIME^MERGE then
        remove the cluster c_i from the active cluster set C
    end
end
return C
```

Like related work, the WAC algorithm can merge clusters but not split them. Splitting might be beneficial when the algorithm detects that a cluster contains multiple distinct stories and should be split into multiple groups. However, splitting was not considered in this research and is left as a potential future research direction.

5.2.2 Implementation details

This section presents the details and decisions made during the algorithm development. It contains information on the multilingual language model used for generating the articles' content embeddings, the named entity recognition model used to extract named entities from the articles, the hyper-parameters used when calculating the Wasserstein distance, and the value sets used for finding the optimal hyper-parameters. The code is implemented using PyTorch [195].

Language model

The article embeddings are created using a pre-trained SBERT [240] based on the XLM-RoBERTa model [22], specifically the `paraphrase-merging-mpnet-base-v2` model[1]. The language model consists of 278M parameters and was pre-trained on parallel data from 50+ languages. The model uses the SentencePiece tokenizer, with a vocabulary size of 250k tokens.

Named entity recognition model

The article's named entities are extracted using the multilingual NER model, named `span-marker-mbert-base-multinerd`[2], created using SpanMarker [242] model and trained on the MultiNERD [243] data set. The model supports 10 languages: English, German, Spanish, French, Italian, Dutch, Polish, Portuguese, Russian, and Chinese, covering our evaluation's main languages. The label set the selected NER model detects includes people, organizations, locations, animals, events, temporal values, and more.

[1] `https://huggingface.co/sentence-transformers/paraphrase-merging-mpnet-base-v2`
[2] `https://huggingface.co/tomaarsen/span-marker-mbert-base-multinerd`

Wasserstein distance parameters

We compute the Wasserstein distance using the Sinkhorn-Knopp algorithm [63]. This algorithm requires two hyper-parameters: (1) regularization factor γ, and (2) the number of iterations *nit*. The regularization factor controls the precision of the final Wasserstein distance. Smaller factors yield more precise solutions but also increase the risk of encountering under- or over-flowing gradient values. Following prior research [192], we conducted experiments with regularization factor values selected from $\{0.01, 0.1, 1\}$, ultimately settling on $\gamma = 0.1$ as it consistently produced stable results. Furthermore, we experimented with different values for *nit* taken from the set $\{5, 10, 20, 50, 100\}$ and found that the Sinkhorn-Knopp algorithm already converges when $nit = 10$, which is the value used in our experiments.

Hyper-parameter selection

The final hyper-parameters for article clustering and cluster merging were determined through a grid search conducted on the training data set. The parameters considered for article clustering included whether to perform monolingual or multilingual article clustering, the rank threshold ($\tau_{\text{RANK}}^{\text{CLUSTER}}$) with values in the set $\{0.4, 0.5, 0.6, 0.7\}$, the named entity threshold ($\tau_{\text{ENTS}}^{\text{CLUSTER}}$) taken from $\{0, 0.2, 0.4, 0.6\}$, the temporal threshold ($\tau_{\text{TIME}}^{\text{CLUSTER}}$) taken from $\{0.1, 0.2, 0.3, 0.4\}$, and the time scale value in days (σ_{CLUSTER}) with values in the set $\{1, 2, 3, 4, 5\}$. For the cluster merging parameters, we explored the rank threshold ($\tau_{\text{RANK}}^{\text{MERGE}}$) with values taken from $\{0.6, 0.7, 0.8, 0.9\}$, the temporal threshold ($\tau_{\text{TIME}}^{\text{MERGE}}$) taken from $\{0.1, 0.2, 0.3, 0.4\}$, the time scaling in days (σ_{MERGE}) where the values were from the set $\{1, 2, 3\}$. The optimal parameters are then used in the evaluation.

5.2.3 Experiment setting

We will now outline the experimental setup, describing the datasets and how they are prepared for the experiments. Next, we will detail the evaluation metrics. Finally, we will introduce the algorithms being compared.

Data sets

We utilized the data sets discussed in Section 5.1 to evaluate the algorithm's performance. This includes the CDET data set, prepared by Miranda et al. [114], and the newly created OG2021 data set. The statistical details of the CDET data set can be found in Section 5.1.4, while the statistics for the OG2021 are presented in Section 5.1.3.

CDET. In this experiment, we utilized the training portion of the data set to determine the algorithm's hyper-parameters, focusing on article clustering and cluster merging thresholds. After determining these hyper-parameters, we applied the algorithm to the test data set.

OG2021. To assess the algorithm's performance on this data set, we conducted a grid search across the hyper-parameters and reported the best-performing results on the 2021 Tokyo Olympics data set.

Evaluation metrics

Similar to related work [114], [137], [138], we report the standard F1 score and the BCubed F1 score [244], as well as their associated precision and recall scores.

Standard F1 score. Let tp be the number of correctly clustered-together article pairs, fp the number of incorrectly clustered-together article pairs, and fn the number of incorrectly not-clustered-together article pairs. Then we report precision as $P = \frac{tp}{tp+fp}$, recall as $R = \frac{tp}{tp+fn}$. The F1 score is calculated as the harmonic mean of the precision and recall: $F_1 = 2\frac{P \cdot R}{P+R}$. While precision describes how homogenous the clusters are, recall tells us the ratio of article pairs that are correctly clustered together in the set of all article pairs that should be clustered together.

BCubed F1 score. The BCubed F1 score, an extension of the F1 score, evaluates the quality of generated clusters using the given article labels. For an article, BCubed precision is calculated as the proportion of articles in its cluster that share the same label, including itself. The overall BCubed precision is the average of these individual precisions across all articles in the data set. Similarly, BCubed recall for an article is calculated as the proportion of articles with the same label in the entire data set that appears in its cluster. Again, the overall BCubed recall is the average of these individual recall scores. Finally, the BCubed F1 score is computed as the harmonic mean of BCubed precision and BCubed recall. BCubed metrics generally favor solutions that make mistakes in clusters already containing many mistakes and make errors in larger clusters rather than smaller ones.

Comparing algorithms

This section introduces the algorithms used to compare with the proposed approach. The selected algorithms are multilingual news clustering algorithms, a category with limited options. Since the algorithms were already applied to the data set used in this experiment, we report the performance metrics found in their original papers.

Tau-search. This algorithm conducts multilingual news stream clustering in two steps [114]. Initially, it generates monolingual event clusters using language-specific features, including TF-IDF subvectors of words, word lemmas, and named entities extracted from the articles. These clusters are then merged into multilingual clusters using cross-lingual word embeddings. Their best-performing multilingual model evaluates distances by comparing every language only to English, and cross-lingual clustering decisions are based solely on these calculated distances.

NewsLens*. This algorithm builds upon the existing newsLens system [132] by processing articles in batches, identifying monolingual local topics to link articles across time and languages [137]. Modifications include a replaying strategy to link monolingual local topics into events and fine-tuning an SBERT language model for creating multilingual clusters.

MNCPS. This algorithm aims to minimize reliance on language-specific features [138]. It obtains article representations using an SBERT language model, computes the best-ranked cluster, decides if the article is accepted into the best-ranked cluster, enters it accordingly, and merges clusters related to the same event. The algorithm uses Rank-SVM models [245] to rank clusters based on the article's publication datetime and content embeddings and determine which clusters to merge.

MSC. Similar to the proposed WAC algorithm, this one uses named entity recognition models for extracting relevant information and multilingual language models to create a multilingual content representation [246]. The articles are then compared in a multilingual vector space by measuring the entity, content and temporal similarity between the article

and each event. The article is assigned to the event if all similarities exceed the predefined thresholds. We report the metrics as stated in the original paper.

5.2.4 Experiment results

This section presents the experiment results. We first provide an overview of the algorithm's performance, followed by an analysis of the algorithm's hyper-parameters and components. Throughout the analysis, we compared three different variants of the original WAC implementation, where the differences are only in the article clustering algorithm:

- $\text{WAC}_{\text{MONO+NER}}$: conducting monolingual article clustering and utilizing entity similarity score,
- WAC_{MONO}: conducting monolingual article clustering without utilizing entity similarity score, and
- $\text{WAC}_{\text{MULTI}}$: conducting multilingual article clustering without utilizing entity similarity score.

All three variants use the same cluster merging approach.

Clustering performance analysis

Depending on the data set used, we chose different hyper-parameter values for the WAC algorithm. Additionally, we report the performance of the comparison algorithms, which enabled us to retrieve or compute the results for each dataset.

CDET. The experiment results performed on the CDET data set can be found in Table 5.8. We exclusively present the best performance scores among all variations, which correspond to the $\text{WAC}_{\text{MULTI}}$ variant with the following article clustering hyper-parameters: performing multilingual article clustering, the rank threshold $\tau_{\text{RANK}}^{\text{CLUSTER}} = 0.5$, no named entity score used, i.e. $\tau_{\text{ENTS}}^{\text{CLUSTER}} = 0$, the time threshold $\tau_{\text{TIME}}^{\text{CLUSTER}} = 0.3$, and time scale $\sigma_{\text{CLUSTER}} = 3$ days. For the cluster merging algorithm, the hyper-parameters are as follows: rank threshold $\tau_{\text{RANK}}^{\text{MERGE}} = 0.7$, time threshold $\tau_{\text{TIME}}^{\text{MERGE}} = 0.3$, and time scale $\sigma_{\text{MERGE}} = 3$ days. These parameters were selected based on the CDET training data set analysis.

Table 5.8: The performance results on the CDET data set (Standard and BCubed metrics reported). The bold and underlined values represent the best and second-best performances, respectively.

	Standard			BCubed			
Algorithm	F1 score	Precision	Recall	F1 score	Precision	Recall	Clusters
CLASSIFICATION-BASED							
Tau-search	84.00	83.00	85.00	-	-	-	-
NewsLens*	86.49	85.11	<u>87.92</u>	82.06	80.25	<u>83.97</u>	606
MNCPS	**97.21**	<u>97.01</u>	**97.42**	**90.10**	<u>89.70</u>	**90.51**	812
DISTANCE-BASED							
MSC	72.20	79.70	66.00	-	-	-	-
$\text{WAC}_{\text{MULTI}}$	<u>92.20</u>	**98.55**	86.62	<u>86.67</u>	**92.94**	81.20	1066

The results indicate that the proposed WAC approach demonstrates comparable performance to the classification-based algorithms concerning F1 score and precision and outperforms the algorithm MSC. However, the recall scores in standard and BCubed metrics are smaller than those of most compared algorithms, suggesting that our algorithm could improve grouping articles describing the same event. Additionally, the WAC approach exhibits the highest standard and BCubed precision, indicating that the generated clusters predominantly consist of relevant news articles.

OG2021. The results of the experiments conducted on the 2021 Tokyo Olympics data set can be found in Table 5.9. We compare all three WAC variations and report the results that achieved the best performance. The hyper-parameter values were selected based on the hyper-parameter analysis conducted on the entire data set. Additionally, we present the performance scores when all articles are grouped into a single cluster, denoted as $\text{Clusters}_{n=1}$, and when each article is assigned to its cluster, denoted as $\text{Clusters}_{n=10,940}$.

Table 5.9: The performance results on the OG2021 data set (Standard and BCubed metrics reported). The bold and underlined values represent the best and second-best performances, respectively.

	Standard			BCubed			
Algorithm	F1	P	R	F1	P	R	Clusters
EDGE CASES							
$\text{Clusters}_{n=1}$	1.01	00.51	100.00	1.03	00.52	100.00	1
$\text{Clusters}_{n=10,940}$	00.00	00.00	00.00	21.97	100.00	12.34	10,940
DISTANCE-BASED							
MSC	15.62	37.77	9.85	51.05	**77.78**	37.99	3,901
WAC_{MONO}	<u>33.89</u>	60.79	**23.50**	<u>58.15</u>	65.35	<u>52.37</u>	1,731
$\text{WAC}_{\text{MONO+NER}}$	32.23	<u>61.30</u>	21.86	58.10	65.74	52.04	1,721
$\text{WAC}_{\text{MULTI}}$	**34.03**	**62.55**	23.37	**58.73**	<u>66.48</u>	**52.60**	1,720

The results show that the WAC variants have similar performances and generate a similar number of event clusters. Additionally, the entity similarity score utilized in $\text{WAC}_{\text{MONO+NER}}$ does not appear to improve the algorithm's performance. All three WAC variations outperform the MSC algorithm, suggesting that more advanced methods are needed for assessing the similarity between articles and event clusters.

Hyper-parameter analysis

To evaluate the impact of different hyper-parameters on the performance of the WAC algorithm, we conducted a grid search across the hyper-parameter values. We reported the best performances on the CDET data set in Table 5.10 and the OG2021 data set in Table 5.11. We provide the article clustering hyper-parameters for each WAC variant, specifically if conducting monolingual article clustering, the rank threshold, the entity score threshold, and the time scale. We also include the cluster merging hyper-parameters, specifically the rank threshold and the time scale. The performance results were measured using the standard and BCubed F1 scores, precision, recall, and the number of clusters generated.

CDET. The results show that the $\text{WAC}_{\text{MULTI}}$ variant performs best. Adding the entity similarity score method to the algorithm generally reduces both the standard and BCubed

F1 and recall scores. This can signify that entity matching can be too strict of a condition when grouping articles. In addition, increasing the article clustering parameter's rank score threshold generally decreases the algorithms' standard and BCubed F1 scores by more than 4%.

Table 5.10: The WAC parameter analysis results on the CDET data set (Standard and BCubed metrics reported). The bold and underlined values represent the best and second-best performances, respectively.

		Article Clustering				Cluster Merging			Standard			BCubed			
Variant name	Monolingual	rank th.	ents th.	time th.	time scl.	rank th.	time th.	time scl.	F1 score	Precision	Recall	F1 score	Precision	Recall	Clusters
WAC$_{\text{MONO}}$	TRUE	0.5	-	0.3	3 days	0.7	0.3	3 days	87.00	98.45	77.95	85.42	93.04	78.95	1066
	TRUE	0.6	-	0.3	3 days	0.7	0.3	3 days	69.50	98.71	53.63	81.08	**94.14**	71.20	1108
WAC$_{\text{MONO+NER}}$	TRUE	0.5	0.2	0.3	3 days	0.7	0.3	3 days	85.02	98.52	74.77	84.78	93.51	77.54	1089
	TRUE	0.6	0.2	0.3	3 days	0.7	0.3	3 days	67.23	98.12	51.14	79.72	93.80	69.32	1109
WAC$_{\text{MULTI}}$	FALSE	0.5	-	0.3	3 days	0.7	0.3	3 days	**92.20**	98.55	**86.62**	**86.67**	92.94	**81.20**	1074
	FALSE	0.6	-	0.3	3 days	0.7	0.3	3 days	74.43	**98.81**	59.70	81.98	94.00	72.68	1112

OG2021. Similar observations can be drawn from the OG2021 data set results. Increasing the rank score threshold for the article clustering generally diminishes the algorithm's performance. Additionally, incorporating the entity similarity score method does not enhance the algorithm's performance; instead, it appears to have a detrimental effect. Moreover, the best performance is achieved by the WAC$_{\text{MULTI}}$ variants.

Table 5.11: The WAC parameter analysis results on the OG2021 data set (Standard and BCubed metrics reported). The bold and underlined values represent the best and second-best performances, respectively.

		Article Clustering				Cluster Merging			Standard			BCubed			
Variant name	Monolingual	rank th.	ents th.	time th.	time scl.	rank th.	time th.	time scl.	F1 score	Precision	Recall	F1 score	Precision	Recall	Clusters
WAC$_{\text{MONO}}$	TRUE	0.7	-	0.3	1 day	0.9	0.3	1 day	26.58	**78.73**	15.99	59.09	**83.29**	45.79	2,953
	TRUE	0.7	-	0.3	3 days	0.9	0.3	1 day	26.39	76.25	15.96	58.60	82.65	45.39	2,933
	TRUE	0.7	-	0.3	1 day	0.85	0.3	1 day	33.89	60.79	23.50	58.15	65.35	52.36	1,731
	TRUE	0.7	-	0.3	3 days	0.85	0.3	1 day	32.10	56.72	22.38	57.85	65.36	51.88	1,746
	TRUE	0.8	-	0.3	1 day	0.9	0.3	1 day	24.92	78.53	14.81	58.07	83.10	44.62	2,946
	TRUE	0.8	-	0.3	3 days	0.9	0.3	1 day	24.99	78.52	14.86	58.13	83.08	44.71	2,947
	TRUE	0.8	-	0.3	1 day	0.85	0.3	1 day	33.01	61.73	22.53	58.09	65.48	52.19	1,717
	TRUE	0.8	-	0.3	3 days	0.85	0.3	1 day	33.09	61.95	22.57	58.08	65.57	52.12	1,720
WAC$_{\text{MONO+NER}}$	TRUE	0.7	0.8	0.3	1 day	0.9	0.3	1 day	24.87	78.02	14.79	58.18	82.88	44.79	2,950
	TRUE	0.7	0.8	0.3	3 days	0.9	0.3	1 day	24.19	77.32	14.34	58.21	83.20	44.77	2,954
	TRUE	0.7	0.8	0.3	1 day	0.85	0.3	1 day	32.12	60.87	21.81	58.00	65.78	51.87	1,721
	TRUE	0.7	0.8	0.3	3 days	0.85	0.3	1 day	31.10	61.69	20.79	58.15	66.65	51.57	1,731
	TRUE	0.8	0.8	0.3	1 day	0.9	0.3	1 day	24.60	78.17	14.60	57.97	83.06	44.52	2,942
	TRUE	0.8	0.8	0.3	3 days	0.9	0.3	1 day	24.80	77.77	14.76	58.08	83.09	44.65	2,942
	TRUE	0.8	0.8	0.3	1 day	0.85	0.3	1 day	32.26	61.31	21.89	57.98	65.46	52.03	1,718
	TRUE	0.8	0.8	0.3	3 days	0.85	0.3	1 day	32.23	61.30	21.86	58.10	65.74	52.04	1,721
WAC$_{\text{MULTI}}$	FALSE	0.7	-	0.3	1 day	0.9	0.3	1 day	28.10	77.09	17.19	59.36	82.97	46.21	2,952
	FALSE	0.7	-	0.3	3 days	0.9	0.3	1 day	28.47	77.14	17.45	**59.37**	82.86	46.26	2,930
	FALSE	0.7	-	0.3	1 day	0.85	0.3	1 day	33.39	57.56	**23.51**	58.06	64.86	52.55	1,735
	FALSE	0.7	-	0.3	3 days	0.85	0.3	1 day	**34.03**	62.55	23.37	58.73	66.48	**52.60**	1,748
	FALSE	0.8	-	0.3	1 day	0.9	0.3	1 day	24.92	78.53	14.81	58.07	83.10	44.62	2,946
	FALSE	0.8	-	0.3	3 days	0.9	0.3	1 day	24.99	78.52	14.86	58.13	83.08	44.71	2,947
	FALSE	0.8	-	0.3	1 day	0.85	0.3	1 day	33.01	61.73	22.53	58.09	65.48	52.19	1,717
	FALSE	0.8	-	0.3	3 days	0.85	0.3	1 day	33.09	61.95	22.57	58.08	65.57	52.12	1,720

Cluster merging assessment analysis. To evaluate the cluster merging process's impact on the algorithm's performance, we compare the WAC algorithm variants to those where the cluster merging phase was not performed. We focus solely on WAC$_{\text{MULTI}}$ variant, as it already generates multilingual clusters during the article clustering phase. The analysis results are presented in Table 5.12.

The comparison results show that the algorithm's performance drops drastically when cluster merging is not executed. It also generates more clusters, reflected in the lower recall

scores. This indicates that the articles describing the same story are distributed across multiple clusters. Therefore, the results show that the cluster merging step is essential for generating good-quality article clusters in our algorithm.

Table 5.12: The cluster merging assessment analysis on the CDET data set (Standard and BCubed metrics reported). The bold and underlined values represent the best and second-best performances, respectively.

		Article Clustering				Cluster Merging			Standard			BCubed			
Variant name	Monolingual	rank th.	ents th.	time th.	time scl.	rank th.	time th.	time scl.	F1 score	Precision	Recall	F1 score	Precision	Recall	Clusters
WAC_{MULTI}	FALSE	0.5	-	0.3	3 days	0.7	0.3	3 days	**92.20**	98.55	**86.62**	**86.67**	92.94	**81.20**	1074
$WAC_{MULTI/NO\ MERGE}$	FALSE	0.5	-	0.3	3 days	-	-	-	56.04	98.71	39.12	71.14	96.98	56.17	2339
WAC_{MULTI}	FALSE	0.6	-	0.3	3 days	0.7	0.3	3 days	74.43	98.81	59.70	81.98	94.00	72.68	1112
$WAC_{MULTI/NO\ MERGE}$	FALSE	0.6	-	0.3	3 days	-	-	-	24.28	**99.40**	13.83	47.10	**99.04**	31.59	4675

5.3 Discussion

In this chapter, we highlight the importance of using neural network-based approaches to support text stream clustering approaches and data sets. The novel 2021 Tokyo Olympics data set comprises news articles in various languages, detailing the events during the 2021 Tokyo Olympics, where articles that report on the same event share a common label. The data set was compared to an existing one, showing the commonalities and differences between them. Afterwards, the two data sets were used to evaluate WAC, a novel unsupervised news stream clustering algorithm. The experiment results show that WAC is comparable, if not superior, to the selected comparison algorithms. The following discussion will delve into the findings concerning both the created data set and the WAC algorithm.

5.3.1 Discussion on news clustering data sets comparison

In this section, we discuss the data set creation process and the novel 2021 Tokyo Olympics data set, including their advantages and disadvantages and theoretical and practical implications.

Creation process analysis

The creation of the 2021 Tokyo Olympics data set depended on several key components. Firstly, the article retrieval process involved determining the time range, languages, and concepts relevant to the desired articles. While the time range was specifically set to cover the selected event duration, the languages and concepts could be expanded to include more sports disciplines and media coverage in various languages.

Next, the selection of the automatic news stream clustering determined the initial quality of the article clusters. An alternative approach could involve clustering news articles for each language separately and then joining the clusters automatically or during the manual evaluation and annotation phase. This approach would result in articles having two labels, one corresponding to the clusters in the monolingual setting and the other for the multilingual setting. However, merging multiple clusters may lead to unintuitive matching, where multiple monolingual clusters of the same language are mapped into a single multilingual cluster, raising questions about why the monolingual clusters are not joined also in the monolingual setting.

Finally, articles were manually merged into clusters based on the predefined criteria. To create a cleaner data set, the criteria suggested the removal of articles resembling

"click-bait" and containing schedules about different events. However, this type of news is commonly encountered in real-world scenarios. Removing such articles from the data set skews the distribution and makes it less representative of the actual distribution of articles in the wild. Therefore, labelling such articles with a special tag could help the data set retain its initial distributions while providing information about the nature of the articles.

Multi-label news clustering

Identifying the correct event cluster for a particular news article may pose challenges due to the possibility of an article describing multiple events simultaneously. In such cases, it could be beneficial to create a multi-label news stream clustering data set, allowing news articles to be associated with multiple event clusters. This approach enables a more granular evaluation of news stream clustering algorithms. However, it is important to note that generating such a data set would require additional time and resources.

To address this challenge, the development of novel data set creation methods could prove helpful. These methods could help streamline the process, reduce the resource requirements, and contribute to the creation of more diverse data sets for text stream clustering.

5.3.2 Discussion on online news article clustering

In this section, we discuss the WAC algorithm's advantages and disadvantages. We explore its strengths and weaknesses compared to the selected algorithms, considering theoretical and practical implications.

Algorithm performance

On the CDET data set, the proposed WAC algorithm performs comparably to the classification-based clustering algorithms, achieving an approximate 92% Standard F1 score and 87% BCubed F1 score. While it boasts the best precision in both standard and BCubed metrics, it exhibits the lowest recall among the compared algorithms. This is reflected in a higher number of clusters, generating 1,066 clusters instead of the 394 clusters in the test data set.

On the 2021 Tokyo Olympics data set, the $\text{WAC}_{\text{MULTI}}$ algorithm variation performs the best with a 34% Standard F1 score and 58% BCubed F1 score. However, the differences in performance across all variations are negligible. In addition, the WAC's performance drops by approximately 53% in the Standard F1 score and 27% in the BCubed F1 score when compared to its performance on the CDET data set. This suggests that the 2021 Tokyo Olympics data set is more challenging for appropriate multilingual article clustering, making it a valuable benchmark data set for future text stream clustering algorithms.

The performance of the algorithm on the CDET data set suggests that it is not necessary to employ classification models for measuring cross-lingual similarity between sets of articles. This can be attributed to the effectiveness of the multilingual vector representations generated by the SBERT language models, proven to be valuable in tasks involving cross-lingual and multilingual semantic text similarity. Additionally, the Wasserstein distance enables the comparison of clusters containing a non-equal number of articles. Therefore, combining these two elements might serve as a substitute for the classification models traditionally used in measuring cross-lingual similarity between article clusters.

Hyper-parameter analysis

The results indicate that the WAC algorithm performs best when performing multilingual article clustering in Algorithm 5.4, again suggesting that the generated content vector representations are appropriate for cross-lingual text comparison. Additionally, increasing the article clustering rank threshold decreases the algorithm's performance on the CDET data set. This suggests that higher rank thresholds may be too stringent, leading to inadequate article clustering.

The use of the entity score function defined in Algorithm 5.3 does not contribute to the algorithm's performance improvement. This observation may be attributed to the SBERT language model used, as it already encodes information about the entities found in the text into the content embeddings. Additionally, named entities can take various forms, making direct comparisons more challenging. Incorporating named entity disambiguation methods could potentially address this issue by determining which entities are the same.

Cluster merging assessment analysis. The cluster merging assessment analysis shows that the proposed cluster merging greatly improves the WAC algorithm's performance. Performance increases by up to 50% standard F1 score and 34% BCubed F1 score, depending on the article clustering algorithm's rank threshold selection. Furthermore, the cluster merging algorithm shows that the Wasserstein distance is a viable metric for assessing which article clusters to merge.

However, computing the Wasserstein distance gets more expensive as the number of clusters and their size increase. Because of this, the Wasserstein distance is more appropriate for scenarios where the news cluster size is usually smaller. To alleviate this issue, one can limit the number of articles used to compute the distance by utilizing just a sample of articles from the cluster. This would reduce the computational resources required to compare and determine if two clusters should be merged. We leave these experiments for future work.

Chapter 6

Streaming Multi-Modal Data Fusion

This chapter focuses on our research on streaming multi-modal data fusion approaches. Section 6.1 introduces the formal definition of streaming data fusion in the Internet of Things domain. Section 6.2 proposes extending the formal definition presented in the previous section by incorporating multi-modal data sources, including text data, and showcasing its practical implementation on stock market predictions. Finally, Section 6.3 discusses the experiment results and the advantages and disadvantages of the work presented in this chapter.

Section 6.1 is based on the published paper [247], while Section 6.2 is research work that is not published yet. This chapter introduces two scientific contributions: [SC-4] an extension of the formal definition of data fusion of heterogeneous streaming data sources, which includes text data, and a method for using texts for data stream predictions, and [SC-5] evaluation of the proposed methods on various data sources and real-world scenarios.

6.1 Defining Streaming Data Fusion for the Internet of Things

The scientific community has been discussing the rising amount of data originating from the Internet of Things (IoT) for more than a decade. The IoT reached the mass market in early 2014, and its ubiquitous influence and challenges are still permeating the scientific literature. The field of Big Data processing has improved drastically, and a plethora of solutions for various IoT problems have reached their production stage [248].

The volume of the data keeps rising, and as the technology penetrates new markets, the industry and academia are being faced with new challenges. The need for efficient and accurate analysis of these data is still an issue [249]. Stream processing [168] has been established as a potential answer to the analysis of Big Data, and incremental learning has been rediscovered to answer some of the challenges (like concept drift [250] or learning efficiency [251]). While the field of incremental learning has matured through the last decade and various algorithms have been described, tested and implemented in various software libraries, the applications of methodologies from a laboratory to the real world have been scarce. Throughout our work in various applications within the environmental domain, water management, traffic, energy efficiency and smart-grid modelling, we have identified the following shortcomings: (1) the most comprehensive software library [252] for stream mining methods is an academic project and, therefore, requires an additional effort when migrating to the production, (2) modern stream mining frameworks (like Apache Spark, Flink and others) do not implement state-of-the-art incremental learning methodologies or online data fusion strategies; it is also extremely difficult to find an operational implementation of an advanced incremental learning regression, and (3) most of the scientific work on incremental learning has taken place inside the lab, emulating unreal (ideal) con-

ditions that are rarely encountered in the real world; this remark mostly applies to the data preparation step (including data fusion and generation of machine-learning-ready rich data streams).

The lack of online data pre-processing techniques also reduces the possibility of using hybrid approaches, where data pre-processing is done online, and prediction models are implemented using traditional machine learning (ML) approaches. McKinsey has established that up to 40% of the data value emerging from the IoT is hidden within the synergy effects of different systems [253]. Except for the ISDI framework [254], which solves time alignment issues of data integration, a generic methodology for generating feature vectors for machine learning approaches in the IoT scenario does not exist yet.

The main research objective in this section is to formally define data fusion of heterogeneous streaming data sources for the IoT domain, which can be used to describe different streaming data fusion frameworks.

We propose a formal definition of streaming data fusion, which includes the time component when describing the sequence of observations in a data stream. This definition includes functions for aggregating data stream observations, sampling values at consistent time intervals, and merging multiple data streams. Additionally, it defines forecast values that the forecasting model needs to predict and the feature vector, which contains data crucial for modelling a specific system. This definition is used to describe a streaming data fusion framework [247]. Since this section of the thesis is dedicated to the definition of streaming data fusion, we do not include the framework's description but direct the reader to the cited paper for further details.

The remainder of the section is structured as follows. First, we define the problem in Section 6.1.1 and discuss the different types of data sources in Section 6.1.2. We then describe the components of streaming data fusion: we define the data streams in Section 6.1.3 and the data stream functions in Section 6.1.4. Next, we outline the forecast data stream in Section 6.1.5 and conclude with the definition of the feature vector in Section 6.1.6.

6.1.1 Problem definition

One of the exploitation scenarios for the vast IoT data is to take advantage of its predictive potential through machine learning methods. For example, based on historical data from a smart grid, we can build a model capable of predicting energy consumption profiles for the next day. This will help to plan the energy distribution better and thus provide cheaper energy for the end user. To create the best possible predictions, we need to be able to work not only with the current power consumption values but also with historical data, different stream aggregates and derivatives and, what is even more important, we need to be able to expand the data streams with relevant contextual information, such as weather data, human behavior data and weather forecasts. Moreover, an online algorithm for creating rich feature vectors for machine learning methodologies is needed to provide new feature vectors as soon as possible and to support incremental learning scenarios.

The two dominant reasons why this kind of data fusion task is not trivial are the heterogeneity of the IoT data and its time incoherence.

According to [255], **heterogeneity** is an intrinsic property of Big Data. Many definitions of heterogeneous data can be found in the literature, and there is no common agreement on the definition among various authors. Among the properties that illustrate the issue are: multi-modality of the data (even considering a mixture of continuous and categorical features and structured and unstructured data) and the technical aspects (i.e. the format of the data), rate of independence, concept drift and dynamics of change, and privacy. In this work, we consider data coming from different sources and focus on heterogeneity based on the discrepancies in the time component [256], [257], which is, in

our opinion, the most important in the IoT data streams. It is manifested through the following properties: (1) **sampling frequency**, (2) **time delay**, and (3) **data availability**. Sampling frequency differs from sensor to sensor. Some sensors implement constant sampling frequency. Different sensors within a setup could implement constant but different sampling frequencies. Many sensors implement approximately constant sampling frequency. The readings happen approximately in the prescribed interval, but the reading might be slightly early or late due to different effects. Some sensors might use arbitrary sampling frequencies (i.e. they might only report an event). Time delay is introduced with transmission latencies, legacy systems and privacy/access issues. Measurements might be late from a few milliseconds up to one day (i.e. when data is transmitted from a legacy system via an FTP connection). Delay is closely related to data availability.

A data stream is a sequence of values with a corresponding timestamp (in IoT, a timestamp denotes the time when the measurement was taken). We define a **coherent time series** as a sequence where each subsequent measurement in a series has been taken later. The most obvious data source which breaks time coherence is the weather forecast data stream. The forecasting models update their predictions regularly, usually every hour. With every update, older forecasts are updated with more accurate values based on recent data.

Based on the issues arising from heterogeneity of the data, we define a **harmonic set of data streams**. A harmonic set of data streams consists of a set of data streams, where each stream has a matching sampling frequency (and phase), at least one matching timestamp, and all the data needed to generate a feature vector successfully.

The following sections present different types of data sources, their main characteristics, and a thorough mathematical formulation of the basic concepts and approaches of streaming data fusion.

6.1.2 Types of data sources

Modelling in IoT scenarios is based on three different types of data sources:

Sensor data. Sensor data most often originates from IoT devices but can also be obtained by crawling a particular web resource. The data is usually obtained close to real-time. However, different lags can be introduced for various reasons. The data fusion system should be able to handle these time-related inconsistencies and build correct feature vectors based on the most recent available data.

Weather forecasts. Weather forecasts are available for the future (usually, we need them until our prediction horizon). The forecasts represent an incoherent data stream (as the values update with time), which must be handled appropriately in the stream fusion system.

Static data. This is the data that, by definition, does not change in time. Values are known for the indefinite future. The data includes attributes like day of the week, day in the year, hour of the day, holiday, day before the holiday, working hours, etc. For simplicity reasons, we handle static data as a stream.

Availability of the three different types of data differs as depicted in Figure 6.1. Depending on the delivery mechanisms, sensor data arrives at the stream processing components with different lags. Figure 6.1 also introduces the **available data horizon**, which is the latest timestamp for which all sensor data streams are available. It represents the latest timestamp for which feature vector generation can be triggered. Feature vectors are built

to calculate predictions at the prediction horizon. Static and weather forecast data are, therefore, usually considered for that timestamp within feature vectors.

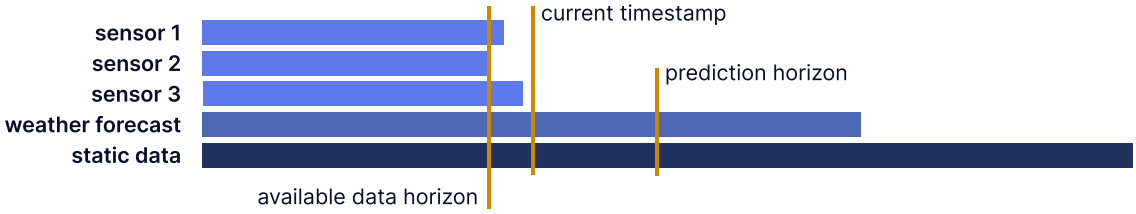


Figure 6.1: Data availability of different types of data sources. Sensor data is delivered in (almost) real-time. However, some legacy systems might introduce longer lags. Weather forecasts are available for a particular time in the future, while static data (i.e. date/time features and human behavior data) are usually always available.

6.1.3 Data streams

In most literature, a data stream is represented by a sequence of values $x_1, x_2, \ldots, x_n$ where $x_i \in \mathbb{R}$ is an observation for all $i \in \{1, 2, \ldots, n\}$. We recognize that this notation has a drawback: it does not contain any information about when a particular value has been provided. Time is an important factor in deciding when to start a specific process on the data stream. To that end, we present a new definition of a data stream that includes time.

A **data stream** $\mathbb{O}_x^n$ is an ordered set of value-time observation pairs provided by a sensor or some other source of data and is described as

$$\mathbb{O}_x^n = \left\{(x_1, t_x^{(1)}), (x_2, t_x^{(2)}), \ldots, (x_n, t_x^{(n)})\right\},$$

where $t_x^{(k)}$ is the time of observation x_k and $t_x^{(i)} \leq t_x^{(j)}$ for all $i \leq j$. A **data stream window** $\mathbb{O}_x^{i,j}$ is a subset which contains observations between the i-th and j-th entries of a data stream $\mathbb{O}_x^n$ and is described as

$$\mathbb{O}_x^{i,j} = \left\{(x_i, t_x^{(i)}), (x_{i+1}, t_x^{(i+1)}), \ldots, (x_j, t_x^{(j)})\right\}.$$

In addition, we will write $\mathbb{O}_x^{1,n} = \mathbb{O}_x^n$.

Remark 6.1. A data stream can also be defined as an ordered set of vector-time observation pairs, e.g. by replacing the values with vectors in the definition above. By doing so, we would allow an observation to contain multiple values and generalize the data stream. For the sake of simplicity, we will use the value-time data stream definition in the rest of the document and will reference this remark when required.

Static data stream

Data streams are dynamic in nature; they are created by retrieving signals from sensors which are then transformed and added to the stream. In some cases, we know what data will come at a particular timestamp. One such case is the *day-of-the-week data stream* where for a given timestamp, we know which day of the week it corresponds to. This type of data streams is defined as **static data streams** $\mathbb{S}_w^n$ and is described as

$$\mathbb{S}_w^n = \left\{(w_1, t_w^{(1)}), \ldots, (w_n, t_w^{(n)})\right\}.$$

Returning to the day-of-the-week static data stream, it contains values $w_i \in [1, 2, \ldots, 7]$ where the number corresponds to a particular day of the week associated with its timestamp (1 corresponding to Monday, 2 to Tuesday, etc.). Another example is the *weekend static data stream*, which contains information on whether a timestamp is in a weekend interval. Its values are $w_i = 1$ if the timestamp $t_w^{(i)}$ is inside a weekend interval and $w_i = 0$ otherwise. Similarly, a *holiday static data stream* is a static data stream which contains information if a timestamp falls in a holiday. Notice that some data streams depend on the context (e.g. culture, country).

6.1.4 Data stream functions

When processing data streams, we might want to manipulate them, sample them and merge multiple data streams to form the final data stream used for time series tasks. To do this, we use different data stream functions, as described in this section.

Data stream aggregate

When we process data streams, we might want to group observations to form a new value that summarizes them. To do this, we require a data stream **aggregate** function that combines the data stream observations and returns the summarized (aggregated) value. A data stream aggregate can also be applied on a data stream window $\mathbb{O}_x^{i,j}$. Generally, a data stream aggregate function is defined as

$$\text{aggr}(\mathbb{O}_x^n) = X,$$

where X is the aggregated value of the provided observations. The most common data stream aggregate functions are (a comprehensive list is available in [258], [259]):

- **Count.** Counts the number of observations in a data stream: $X = n$,
- **Maximum.** Returns the maximum value in a data stream: $X = \max\{x_1, \ldots, x_n\}$,
- **Minimum.** Returns the minimum value in a data stream: $X = \min\{x_1, \ldots, x_n\}$,
- **Sum.** Sums up all values in a data stream: $X = \sum_{i=1}^{n} x_i$.

A more complex example of a data stream aggregate is the **moving average** (MA). This aggregate is used to smooth out short-term fluctuations and highlight longer-term trends. It calculates the average of the observations within a data stream window $\mathbb{O}_x^{i,j}$ and is defined as

$$\text{MA}(\mathbb{O}_x^{i,j}) = \frac{1}{j-i+1} \sum_{k=i}^{j} x_k.$$

When the data stream window shifts, the new MA can be calculated by using the previous MA value:

$$\text{MA}\left(\mathbb{O}_x^{i+1,j+1}\right) = \frac{(j-i+1) \cdot \text{MA}(\mathbb{O}_x^{i,j}) - x_i + x_{j+1}}{j-i+1}$$

A variation of the moving average is the **weighted moving average** (WMA). Let the observations x_k have an associated weight w_k. Then the weighted moving average of the observations within the data stream window $\mathbb{O}_x^{i,j}$ is defined as

$$\text{WMA}(\mathbb{O}_x^{i,j}) = \frac{1}{W^{(i,j)}} \sum_{k=i}^{j} w_k x_k, \quad W^{(i,j)} = \sum_{k=i}^{j} w_k,$$

where $W^{(i,j)}$ is the sum of all weights w_k associated to the observations in the data stream window. The WMA can be updated similarly to the MA as the data stream window shifts.

The second more complex aggregate function is the **exponential moving average** (EMA). It is similar to MA only in that it incorporates a decaying factor, giving the more recent observations greater importance. This aggregate inputs the data stream $\mathbb{O}_x^n$ and is calculated with the following recursive function:

$$\mathrm{EMA}(\mathbb{O}_x^n) = \begin{cases} x_n, & \text{for } n = 1, \\ \alpha(n) \cdot x_n + \big(1 - \alpha(n)\big) \cdot \mathrm{EMA}(\mathbb{O}_x^{n-1}), & \text{for } n \neq 1, \end{cases}$$

where $\alpha(n) = \frac{\Delta t(n)}{T}$, $\Delta t(n) = t_x^{(n)} - t_x^{(n-1)}$ represents the rate of the decay and T is the user defined split time constant.

Once we decide on the aggregate functions to use in processing, we can create an aggregated data stream. An **aggregated data stream** $\mathbb{O}_{x,\mathrm{aggr}}^n$ is a data stream containing the sequence of aggregated values of $\mathbb{O}_x^n$ by using the aggregate function "aggr".

$$\mathbb{O}_{x,\mathrm{aggr}}^n = \Big\{\big(\mathrm{aggr}(\mathbb{O}_x^1), t_x^{(1)}\big), \ldots, \big(\mathrm{aggr}(\mathbb{O}_x^n), t_x^{(n)}\big)\Big\}.$$

We will now look at two aggregated data stream examples:

Moving average data stream. This aggregated data stream is created by using the MA aggregate and is described as

$$\mathbb{O}_{x,\mathrm{MA}}^n = \Big\{\big(\mathrm{MA}(\mathbb{O}_x^{1,k+1}), t_x^{(k+1)}\big), \ldots, \big(\mathrm{MA}(\mathbb{O}_x^{n-k,n}), t_x^{(n)}\big)\Big\},$$

where $k < n$ is the user-defined data stream window size.

Exponential moving average data stream. This aggregated data stream is created by using the EMA aggregate and is described as

$$\mathbb{O}_{x,\mathrm{EMA}}^n = \Big\{\big(\mathrm{EMA}(\mathbb{O}_x^1), t_x^{(1)}\big), \ldots, \big(\mathrm{EMA}(\mathbb{O}_x^n), t_x^{(n)}\big)\Big\}.$$

More complex stream aggregates can require interpolation over the time series and calculation of values, such as the number of extremes in a particular data stream window, the highest n-th derivative in the data stream window, the duration of the largest maximum, etc. Such derivatives are, for example, useful for modelling crop types in earth observation scenarios.

Data stream resampler

One of the properties of data streams is that observations might not come at a constant rate. This can cause problems when multiple data streams need to be synchronized. To handle this issue, we define a **sampling function**, which takes a data stream $\mathbb{O}_x^n$ and a timestamp T as an input and returns a sample value. The function can be described as

$$\mathrm{sampler}(\mathbb{O}_x^n, T) = X_i,$$

where X_i is the sampled value generated from the input parameters. The sample value can be generated using different functions:

- **Last value.** This function returns the last observation that appeared before the provided time T: x_k, where $t_x^{(i)} < T$ for all $i \leq k$ and $T \leq t_x^{(j)}$ for $k < j$.

- **First value.** This function returns the first observation that appears after the provided time T: x_k, where $t_x^{(i)} < T$ for all $i < k$ and $T \leq t_x^{(j)}$ for $k \leq j$.
- **Linear interpolation.** This function returns the sample value by using a linear interpolation between observations around the provided time T, e.g.

$$\text{lin}(\mathbb{O}_x^n, T) = \frac{x_k - x_{k-1}}{t_x^{(k)} - t_x^{(k-1)}}\left(T - t_x^{(k)}\right) + x_k,$$

  where $t_x^{(k-1)} \leq T \leq t_x^{(k)}$.

Once we decide on the sampling function f and a constant time period T, we can create a data stream of sampled data. This type of data stream is defined as a **resampled data stream** containing the sampled values of a provided data stream $\mathbb{O}_x^n$ and is described as

$$\mathbb{O}_{x,\text{sampler}}^m = \{(X_1, T_X^{(1)}), \ldots, (X_m, T_X^{(m)})\},$$

where $X_i = \text{sampler}\left(\mathbb{O}_x^n, T_X^{(i)}\right)$ is the sampled value returned by the sampling function performed at time $T_X^{(i)}$. In addition, the difference of consecutive time values is equal to the constant time period T, i.e.

$$T = T_X^{(i+1)} - T_X^{(i)},$$

for all $i \in \{1, \ldots, m-1\}$.

Data stream merger

Sometimes, we must merge two or multiple data streams into a single one. We define a merger function that can do just that. Suppose we have two data streams $\mathbb{O}_x^n$ and $\mathbb{O}_y^m$ where $t_x^{(i)} = t_y^{(i)}$ for all $i \leq \min\{n, m\}$. A **merger function** takes $\mathbb{O}_x^n$ and $\mathbb{O}_y^m$ as an input and returns the merged data stream, i.e.

$$\text{merger}(\mathbb{O}_x^n, \mathbb{O}_y^m) = \left\{(x_1, y_1, t^{(1)}), \ldots, (x_k, y_k, t^{(k)})\right\},$$

where $k = \min\{n, m\}$ and $t^{(i)} = t_x^{(i)} = t_y^{(i)}$ for all $i \in \{1, \ldots, k\}$. If we adopt the view provided in remark 6.1, the output of the merger function is a data stream. Indeed, if we write the values x_i and y_i as entries of a vector, then the output will follow the generalized definition of a data stream.

6.1.5 Forecast data stream

Forecast data streams provide forecasted values for the future. The difference between the time of the forecast and the time of the forecast generation is called a *prediction horizon*. A simple forecasting model provides forecasts for a constant prediction horizon and can be described simply by $\mathbb{O}_p^n$, where p_i is the forecasted value and $t_p^{(i)}$ refers to a timestamp in the future. More complex forecasting streams (i.e. weather forecasts) provide predictions for multiple time horizons simultaneously. For example, new weather forecasts are generated every hour for the next 48-hour interval. This implies that a forecast for a particular hour in a day gets updated 48 times during this process. A new data stream window $\mathbb{O}_p^{i,j}$ is generated at forecast generation. Thus, the forecast data stream is defined as

$$\mathbb{F}_x^n = \{(\mathbb{O}_p^{1,k}, t_x^{(1)}), (\mathbb{O}_p^{2,k+1}, t_x^{(2)}), \ldots, (\mathbb{O}_p^{n,k+n-1}, t_x^{(n)})\}, \tag{6.1}$$

where k is the number of time horizons and $\mathbb{O}_p^{i,k+i-1}$ is the data stream window generated at time $t_x^{(i)}$. In the case of $k = 1$, the forecast data stream contains a simple forecasting model, generating only one prediction of a constant prediction horizon.

6.1.6 Feature vector

In machine learning, a *feature vector* is a vector that contains data important for the description and modelling of a particular system. Together with a label of a specific data instance, it is used to train a machine learning model. If the label is unknown, the vector can be used to derive predictions (or other results) from a trained model. An element in the vector is called a *feature* (in some literature, it is referred to as an *attribute*). A feature vector which includes contextually rich information derived from a set of data streams will allow a machine learning model to achieve the best possible results in modelling particular phenomena. Such a vector ϕ_{full} should be the final result of a streaming data fusion framework.

A feature vector ϕ_{full} is a representation of a set of observation, static data and forecast data streams $\mathbb{O}_x^n$, $\mathbb{S}_y^n$ and $\mathbb{F}_z^n$, where $x \in \{x_1, \ldots, x_i\}$, $y \in \{y_1, \ldots, y_j\}$ and $z \in \{z_1, \ldots, z_k\}$. We will focus on feature vectors of the following form:

$$\phi_{\text{full}} = \begin{bmatrix} F^{(1)} \\ F^{(2)} \\ \vdots \\ F^{(q)} \end{bmatrix},$$

where q is the size of the feature vector and each feature $F^{(r)}$ is defined with:

$$F^{(r)} = \begin{cases} x_i^{(j)}, & \text{a (resampled) measurement extracted from } \mathbb{O}_{x_i}^n \\ y_i^{(j)}, & \text{a (resampled) forecast extracted from } \mathbb{F}_{y_i}^n \\ z_i^{(j)}, & \text{a (resampled) forecast extracted from } \mathbb{S}_{z_i}^n \\ X_i^{(j)}, & \text{a (resampled) stream aggregate extracted from } \mathbb{O}_{x_i}^n \\ Y_i^{(j)}, & \text{a (resampled) stream aggregate extracted from } \mathbb{F}_{y_i}^n \\ Z_i^{(j)}, & \text{a (resampled) stream aggregate extracted from } \mathbb{S}_{z_i}^n \end{cases}$$

where $r \in \{1, \ldots, q\}$ is the index corresponding to the feature and the index $j \in \{1, \ldots, n\}$ refers to current or historical values of the measurements or the aggregates. Additionally, each feature vector reflects the state of the observed system at resampled timestamp $T_x^{(n)}$.

The proposed definition of streaming data fusion primarily addresses sources related to the IoT domain, such as sensors, weather forecasts, and static data, which mainly comprise numerical and categorical data values. However, to adapt this definition for additional types of data streams, such as text, we need to expand the definition to include these data types and process them appropriately. Furthermore, some of the data may be in formats that are not optimal for machine learning models to process. In the next section, we will expand the formal definition of data fusion to include text in time series forecasting models, addressing these challenges.

6.2 Including Text into Time Series Forecasting Models

Time series forecasting models often use text data as one-dimensional features. By transforming text data into single features, we can handle it like another time series and merge it with other time series through data fusion. This method is frequently used in financial forecasting, where social media posts and news articles are processed to determine sentiment scores or other factors like the frequency of company mentions in the news. However, simplifying text into a one-dimensional feature may result in the loss of valuable information.

Transformer-based language models can create rich, multi-dimensional vector representations of text. These vectors have been successfully used in various NLP tasks, including text classification, information retrieval, and retrieval-augmented generation (RAG). However, these representations might not directly align with time series forecasting models, often treating each input value as a separate feature. In a text embedding vector, the values must be considered together since an individual value does not typically represent specific information about the text. Integrating these multi-dimensional vectors from language models into time series forecasting could enhance the model's performance.

This section focuses on developing and analyzing methods to integrate rich, multi-dimensional textual representations into time series forecasting models by combining data fusion techniques and neural network components.

The main research objectives are to (1) extend the formal definition of data fusion of heterogeneous streaming data sources by including multi-modal data sources, (2) develop different approaches for integrating text streams into time series forecasting models, and (3) evaluate the impact of these methods on the algorithm's performance.

We propose four extensions to the Long Short-Term Memory (LSTM) model that include components for combining text features and time series. These components include a simple Feed-Forward network, a Convolutional Neural Network (CNN) and an Attention mechanism. The extensions are evaluated on fourteen stock time series from companies in the S&P100, combined with news event information retrieved from Event Registry. We used the coefficient of determination (R^2) to evaluate the forecast accuracy for both the stock close and daily volatility dynamics. The results show that the LSTM extensions can effectively use text features to enhance stock close forecasting performance. However, only the extensions incorporating the Attention mechanism significantly improve the performance of predicting the daily volatility.

The remainder of the section is structured as follows. First, we expand the formal definition of data fusion to include heterogeneous streaming data sources in Section 6.2.1. Next, we describe the various LSTM extensions in Section 6.2.2, followed by their implementation details in Section 6.2.3. The experimental setting is detailed in Section 6.2.4. Finally, we present the experiment results in Section 6.2.5.

6.2.1 Extension of the formal definition of data fusion

This section presents an extension to the formal definition of data fusion defined in Section 6.1.1. The extension focuses on including heterogeneous streaming data sources, where a particular feature is represented as a multi-dimensional vector, as in the case of text embeddings.

General data stream

The initial data fusion definition focuses on data stream $\mathbb{O}_x^n = \left\{\left(x_i, t_x^{(i)}\right)\right\}_{i=1}^n$, where x_i are one-dimensional observations or aggregates, or multi-dimensional vectors where each value of the vector corresponds to a single feature. A feature can often be represented by a multi-dimensional vector. For example, text can be represented by a multi-dimensional vector called an embedding. To clarify these cases, the rest of the section will use the following notation

$$\mathbb{O}_v^n = \left\{\left(\vec{v}_1, t_v^{(1)}\right), \left(\vec{v}_2, t_v^{(2)}\right), \ldots, \left(\vec{v}_n, t_v^{(n)}\right)\right\},$$

where $\vec{v}_k$ is a multi-dimensional vector representation of a feature v retrieved at time $t_v^{(k)}$ and $t_v^{(i)} \leq t_v^{(j)}$ for all $i \leq j$.

Data stream mapper

While the previous definitions, such as data stream aggregate and data stream merger, work with data stream windows, the data stream **mapper** maps individual observations from one feature space to another. Generally, the mapper is represented as a function

$$\text{mapper}(\vec{v}) = \vec{w}, \quad \text{mapper} : \mathbb{R}^k \to \mathbb{R}^\ell,$$

where $\vec{v} \in \mathbb{R}^k$ is a multi-dimensional observation in one feature space mapped into a different feature space representation $\vec{w} \in \mathbb{R}^\ell$. The mapper function can be predefined or learned using machine learning methods. We can create a mapped data stream once we decide on the mapper function. A **mapped data stream** $\mathbb{O}^n_{v,\text{mapper}}$ is a data stream containing the sequence of values from $\mathbb{O}^n_v$ that were mapped using the function "mapper".

$$\mathbb{O}^n_{v,\text{mapper}} = \left\{\left(\text{mapper}(\vec{v}_1), t_v^{(1)}\right), \ldots, \left(\text{mapper}(\vec{v}_n), t_v^{(n)}\right)\right\}.$$

Examples of machine learning-based mapper functions. Let us assume we have a text stream

$$\mathbb{O}^n_{\text{txt}} = \left\{\left(\text{txt}_1, t_{\text{txt}}^{(1)}\right), \left(\text{txt}_2, t_{\text{txt}}^{(2)}\right), \ldots, \left(\text{txt}_n, t_{\text{txt}}^{(n)}\right)\right\},$$

where txt_k is the text retrieved at time $t_{\text{txt}}^{(k)}$. Then, examples of machine learning-based mapper functions include sentiment score extraction and sentence embedding generation.

The **sentiment analysis mapper** (SA) is a function that maps the text txt_k into a one-dimensional feature representing the sentiment score of the text $s_k \in [-1, 1]$. The sentiment score is extracted by sending the text through a pre-trained sentiment extraction model. The mapped data stream using the SA mapper is defined as

$$\mathbb{O}^n_{\text{txt,SA}} = \left\{\left(\text{SA}(\text{txt}_1), t_{\text{txt}}^{(1)}\right), \ldots, \left(\text{SA}(\text{txt}_n), t_{\text{txt}}^{(n)}\right)\right\}.$$

The **sentence embedding mapper** (SE) is a function that maps the text txt_k into a multi-dimensional contextual embedding $\vec{e}_k \in \mathbb{R}^k$. The embedding is created using a pre-trained language model specialized for creating contextual vector representations from texts. The mapped data stream using the SE mapper is then defined as

$$\mathbb{O}^n_{\text{txt,SE}} = \left\{\left(\text{SE}(\text{txt}_1), t_{\text{txt}}^{(1)}\right), \ldots, \left(\text{SE}(\text{txt}_n), t_{\text{txt}}^{(n)}\right)\right\}.$$

Multi-modal data fusion and forecasting

We can now align multi-modal data streams using the defined data stream functions. Given the several data streams, we can (1) transform the observations of the data stream into an appropriate vector representation using the *data stream mapper*, (2) use *aggregate function* to summarize the observations, (3) synchronize and temporally align the data streams using a *sampling function*, and (4) use a *merger function* to merge multiple data streams with temporally-aligned observations.

The aligned observations can then be used to create a *feature vector*, which can, together with a label, be used to train a machine learning model or to derive a prediction. However, some observations, such as text embedding, are multi-dimensional vectors which must be considered as a whole, as individual vector values do not necessarily mean anything. In such cases, we might need to prepare multiple feature vectors containing specific and disjoint observations. When this happens, the forecasting model must be designed to accept the feature vectors as inputs and join them internally. By doing so, we can combine data streams of various data types and use them to predict values in the *forecast data stream*.

6.2.2 Stock market forecasting using news features

We evaluate the extended multi-modal data fusion and forecasting definition by using combined stock data streams and news streams to predict stock market dynamics. This is done by exploring multiple approaches for incorporating text features to enhance the model's forecasting capabilities. Our baseline is the Long Short-Term Memory (LSTM) model [174] that only uses stock data streams. We propose multiple extensions to include text features. Figure 6.2 shows the high-level overview of the models' architectures. This section describes the architectures of the models used to predict the stock market dynamics.

Baseline LSTM. This model is used as a baseline to assess the impact of incorporating news streams on forecasting performance. It inputs stock data and processes it through a single LSTM layer. The LSTM outputs are then concatenated with the original stock data, ensuring that the final layer responsible for predicting the actual time series forecast values includes both the outputs from the LSTM and the original data. Additionally, through experimentation, we found that concatenating these values improves the model's performance compared to using only LSTM outputs.

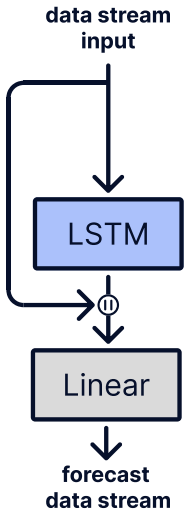
The remaining models are extensions; we will only describe the differences between these extensions and the baseline LSTM model.

$\mathbf{LSTM_{INPUT}}$. In this model extension, the news stream is merged with the data stream before passing through the LSTM layer. Because no other adaption was made to the news stream features, each value in a potential multi-dimensional vector is considered an individual feature.

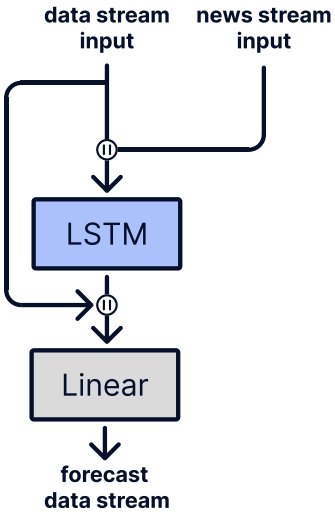
$\mathbf{LSTM_{FNN}}$. This model employs a mapper in the form of a Feed-Forward layer to convert the news stream feature values into multi-dimensional vectors of size `text_output_size`. These vectors are then merged with the data stream input and fed into the LSTM layer. The Feed-Forward layer's purpose is to generate abstract vector representations of the news stream features, aiming to enhance the forecasting capabilities of the model. However, due to the layer's linearity, it is limited to finding relationships within the news features when creating these abstract vector representations.

$\mathbf{LSTM_{CNN}}$. Similar to $\mathrm{LSTM_{FNN}}$, this model starts with a mapper in the form of a convolutional neural network (CNN) to process the news features. The CNN is shallow, comprising 1D convolution (Conv1D) layers with rectified linear unit (ReLU) activation functions. It takes news features as input and outputs vector representations of size `text_output_size`. These vectors are then concatenated with the data stream inputs before being processed through the LSTM layer. The CNN aims to efficiently capture the essential features from the news data using the convolution kernels.

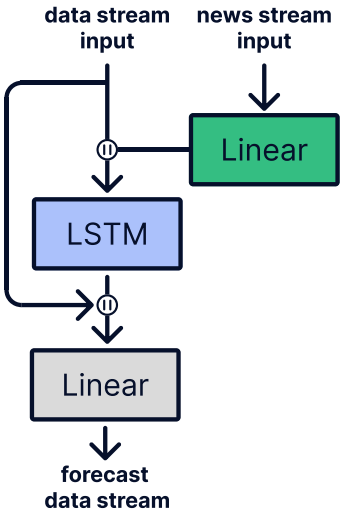
$\mathbf{LSTM_{ATTN}}$. This model incorporates an Attention mechanism to identify which outputs from the data stream LSTM should be emphasized based on features from the news stream. The output from the Attention layer is concatenated with the input values from the data stream before being processed through a linear layer. This linear layer then produces the forecasted data stream. Using the Attention mechanism, the model can identify which data stream LSTM representations are more important for a given inference based on the features from the news stream, which may include information about daily market events.



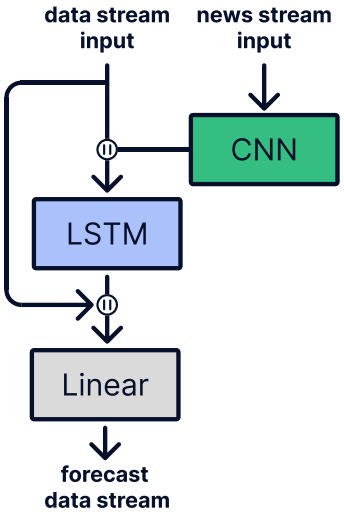
(a) The baseline LSTM. The main method we are comparing with.



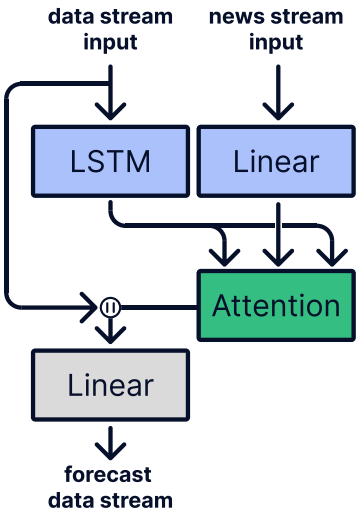
(b) The $\text{LSTM}_{\text{INPUT}}$ method. The text series input is concatenated with the time series features before sending it through the LSTM component.



(c) The LSTM_{FNN} method. The text features are sent through a Feed-Forward network before being concatenated with the time series.



(d) The LSTM_{CNN} method. The text features are sent through a CNN before being concatenated with the time series.



(e) The $\text{LSTM}_{\text{ATTN}}$ method. The text features determine which LSTM representations of the time series should be emphasized.

Figure 6.2: The depiction of the baseline LSTM model and its extensions. The parallel lines in the circle symbol represent the vector concatenation operation.

6.2.3 Implementation details

In this section, we present the details of how we developed the forecasting models. They are implemented using PyTorch [195] and Pytorch Lightning[1]. The source code is available

[1] `https://lightning.ai/docs/pytorch/stable/`

on GitHub [260].

The architecture parameters

All models feature a single LSTM layer. We tested two values of LSTM's `hidden_size`, specifically 32 and 64, where `hidden_size` = 32 provided a better performance on the validation set. The selected `hidden_size` value also determined the output sizes of the intermediary Feed-Forward and CNN layers in the LSTM_{FNN}, LSTM_{CNN}, and $\text{LSTM}_{\text{ATTN}}$ models. The $\text{LSTM}_{\text{ATTN}}$ model utilized a single Attention block to compute the attention outputs. We experimented with various `text_output_size` values, specifically 1, 2, 4, 8, 16, and 32. We found that `text_output_size` = 1 provided the best results and was used across all models.

Model fine-tuning

We updated the model parameters using the Adam optimizer [261] with an initial weight decay of 0.001. The learning rate was determined using Pytorch Lightning's learning rate finder [262], which processes a few batches at different learning rates to find the smallest training loss. For each model, we used the learning rate that the finder identified as the best initial value.

The models were trained using the Mean Absolute Error (MAE) loss function. We included an early stopping criterion: every five epochs, we evaluated the model's performance on the validation set and tracked their loss. If the validation set loss did not improve after ten checks, we stopped the training. We also set an upper limit of 200 epochs for training the models. The models were trained and evaluated on an Nvidia RTX GeForce 3070 8 GB graphics card.

6.2.4 Experiment settings

This section details the experimental setting, including descriptions of the datasets and their preparation, as well as explanations of the evaluation metrics used.

Data sets

This section describes the data set used in the experiment. The dataset includes a data stream of stock market data and a news stream containing information about news events associated with companies related to the stocks.

Stock market data stream. The data streams were retrieved using Yahoo Finance[2], a popular source for financial data offering detailed insights into stock market trends and prices. We used it to collect economic data for companies in the S&P100, which comprises 100 leading U.S. stocks with exchange-listed options. For each company, we gathered data from January 1, 2020, to December 31, 2023, which includes the following daily stock market information:

- *Open.* The price at which the stock started trading on that day.
- *Low.* The lowest stock price within the day.
- *High.* The highest stock price within the day.
- *Close.* The price at which the stock exchange closed for that day.

[2] `https://finance.yahoo.com/`

- *Volume.* The total number of shares traded on that day.

Note that the previous day's closing price may not equal the current day's opening price due to after-hours and pre-market trading activities (i.e., trading that occurs when the U.S. stock exchange is closed) and trading activities in other countries.

News event stream. We retrieve the company's news events using Event Registry [113], a system which collects, clusters and enriches news articles published globally. In our experiments, we utilized these news events (clusters of articles reporting on similar topics) as indicators of significant occurrences related to the company. Each news event is characterized by various attributes, including the event's title, a summary, the number of articles covering the event, and the Wikipedia concepts linked to the event. Additionally, each concept is defined by its unique identifier (corresponding to its Wikipedia URL), a label, and a score that indicates its relevance to the event's title and summary.

For each company in the S&P100, we retrieved their news events published between January 1, 2020, and December 31, 2023, using the company's name as the search parameter (treated as the concept). We limit our retrieval to news events consisting only of English articles. We used the Event Registry API to retrieve the relevant news events.

Data preparation

Stock market data stream. Each one of the 100 retrieved stock market data streams is processed independently. Following previous work [160], [162], [263], we focus on forecasting future close values and daily volatility dynamics.

By forecasting the **future close dynamics**, we focus on the movement of the stock's close value. To do this, we calculate the five-day moving average (MA) of the *close* values, which are then used as the data stream input. To create the forecast data stream, we shift the calculated moving average statistics for one day; the objective of the forecasting model is then to predict the five-day moving average of the close values calculated one day in the future.

$$\mathbb{O}^k_{\mathrm{ds}_{\mathrm{close}}} \mapsto \mathbb{O}^k_{\mathrm{ds}_{\mathrm{close}},\mathrm{MA}} = \mathbb{C}\mathbb{O}^k_{\mathrm{ds}}$$

We define the *daily volatility* as the difference between the *high* and *low* stock price within the day; the greater the difference, the higher the daily volatility is and thus more activity happening on the market. To measure the **stock volatility dynamics**, we compute the five-day moving average (MA) of the daily difference data stream. As the target forecast data stream, we shift the calculated values for one day; thus, the objective is to predict the next day's daily volatility statistics.

$$\mathbb{O}^k_{\mathrm{ds}_{\mathrm{volatility}}} \mapsto \mathbb{O}^k_{\mathrm{ds}_{\mathrm{volatility}},\mathrm{MA}} = \mathbb{V}\mathbb{O}^k_{\mathrm{ds}}$$

News event stream. We process news event data for each S&P100-related company separately. We first filter the news event data sets to retain only the events that either mention the associated company's name or have the company's associated Wikipedia concept of a score greater than 80. This gives us only the news events that are truly associated with the selected company.

Next, we use data stream mappers to convert the raw text from the news stream into vector representations. We utilize both the sentiment analysis mapper and the sentence embedding mapper to do that.

By using the **sentiment analysis mapper** (SA), we extract the sentiment score from the news event's title. We use a distilled version of the RoBERTa [17], [19] model that was

fine-tuned on the financial phrase bank data set[3].

$$\mathbb{O}^m_{\text{ns}} \mapsto \mathbb{O}^m_{\text{ns,SA}} = \mathbb{AO}^m_{\text{ns}}$$

Using the **sentence embedding mapper** (SE), we transform the news event's text, concatenated text and summary into a multi-dimensional feature that contains the contextual information of the event. We use a Sentence-BERT [84] model, specifically, the `paraphrase-multilingual-mpnet-base-v2` model[4], which converts text into a vector space and can be used for tasks related to semantic comparison.

$$\mathbb{O}^m_{\text{ns}} \mapsto \mathbb{O}^m_{\text{ns,SE}} = \mathbb{EO}^m_{\text{ns}}$$

Once we create the vector representation for each news event, we aggregate the mapped data streams by using the **weighted moving average** (WMA) aggregate function to calculate the daily weighted average. We first retrieve all of the news events $\mathbb{O}^{i,j}_{\text{ns,mapper}}$ that reported on an event happening on a particular day. We then calculate the weighted moving average, where weight w_j represents the number of articles covering the event ns_j. We calculate the aggregate for both mapped data streams:

$$\mathbb{AO}^m_{\text{ns}} \mapsto \mathbb{AO}^k_{\text{ns,WMA}}, \quad \mathbb{EO}^m_{\text{ns}} \mapsto \mathbb{EO}^k_{\text{ns,WMA}},$$

where k is the number of unique days between January 1, 2020, and December 31, 2023.

Data stream merging. Once the stock market data stream and news event stream are processed, we use the **data stream merger** function to merge the data streams. We prepare four combinations, two for forecasting the future close dynamics $\mathbb{CO}^k_{\text{ds}}$ and two for stock volatility dynamics $\mathbb{VO}^k_{\text{ds}}$, by merging the corresponding stock market data stream with the sentiment score $\mathbb{AO}^k_{\text{ns,WMA}}$ and sentence embedding $\mathbb{EO}^k_{\text{ns,WMA}}$ news event stream aggregates, respectively. This allows us to test how the addition of news sentiment score and sentence embeddings affects the chosen stock market data forecasting performance.

$$\text{merger}(\mathbb{CO}^k_{\text{ds}}, \mathbb{AO}^k_{\text{ns,WMA}}), \quad \text{merger}(\mathbb{VO}^k_{\text{ds}}, \mathbb{AO}^k_{\text{ns,WMA}}) \tag{6.2}$$

$$\text{merger}(\mathbb{CO}^k_{\text{ds}}, \mathbb{EO}^k_{\text{ns,WMA}}), \quad \text{merger}(\mathbb{VO}^k_{\text{ds}}, \mathbb{EO}^k_{\text{ns,WMA}}) \tag{6.3}$$

These data streams are then used to create the training examples. Each example consists of a sequence of 20 consecutive observations from the merged data stream and a label that is the stock market data stream observation from the next day. We split the data sets into train/validation/test using the 70/10/20 split. In the experiments, we denote the results gathered using the data streams from Equation (6.2) as *sentiment*, and the data streams from Equation (6.3) as *embedding*.

Data set statistics. During post-processing, we measured some of the statistics of the created final data streams. Measuring the news event coverage by day, we found that only fourteen stocks have news events published in over 50% days within the retrieved time window. To assess the performance of models that also use news streams, we focus only on those fourteen stocks to ensure sufficient enough data: Apple (AAPL), AMD (AMD), Amazon (AMZN), Alphabet (GOOG and GOOGL), Intel (INTC), McDonald's (MCD), Microsoft (MSFT), Netflix (NFLX), Nike (NKE), Nvidia (NVDA), Pfizer (PFE), Tesla (TSLA), and Walmart (WMT). These companies span across different domains, including information technology, communication services, consumer discretionary, consumer staples and healthcare. Figure 6.3 shows each company's close time series and news availability over the retrieved time window.

[3] `https://huggingface.co/mrm8488/distilroberta-finetuned-financial-news-sentiment-analysis`

[4] `https://huggingface.co/sentence-transformers/paraphrase-multilingual-mpnet-base-v2`

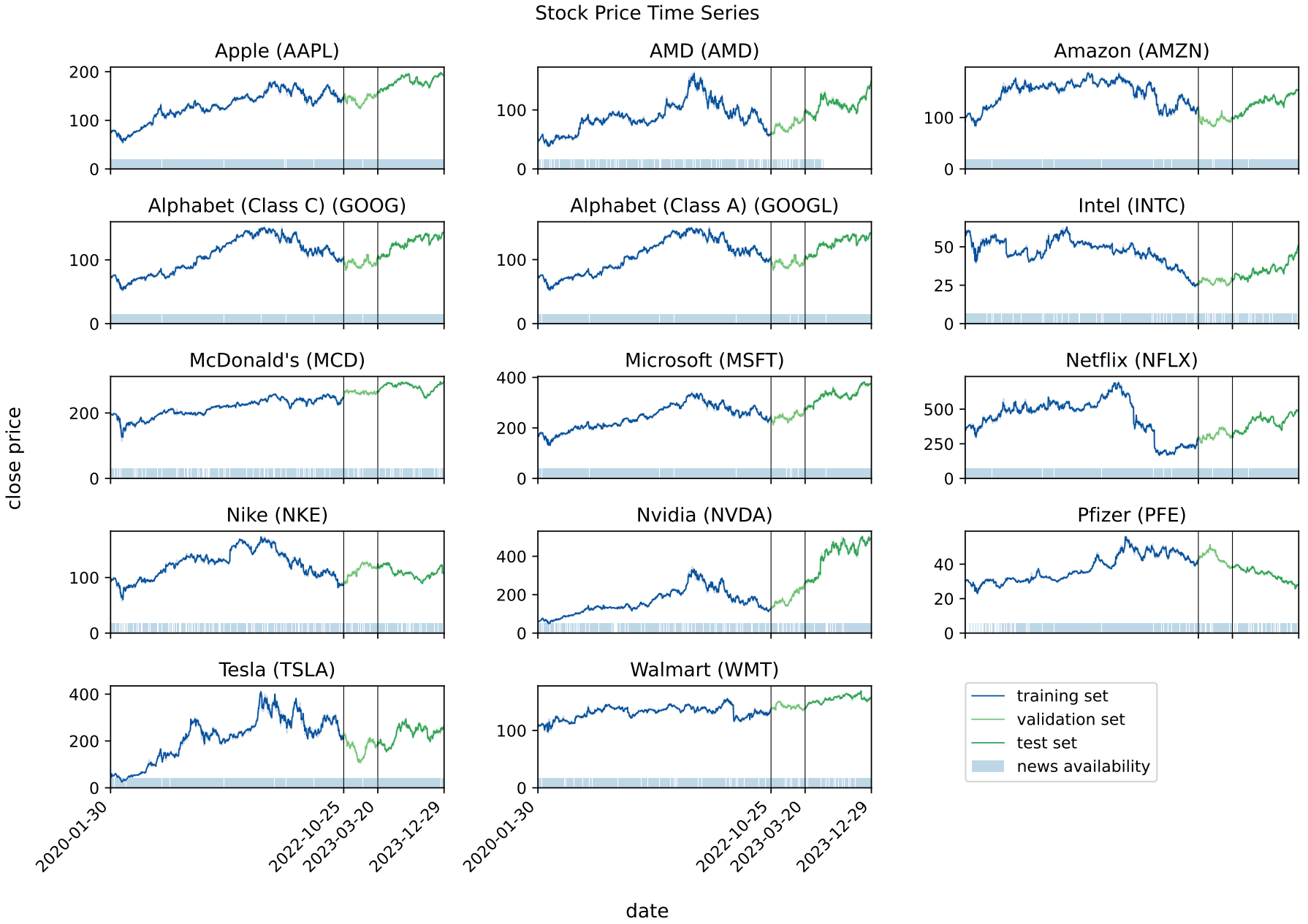


Figure 6.3: The stock price and news availability time series. It shows 14 stock time series separations into the train, evaluation and test data sets and their news availability per day.

Evaluation metrics

Since the stocks' close values are in different scales in the experiments, we use the coefficient of determination to measure the models' forecasting performance.

Coefficient of determination. The R^2 metric is a statistical measure that gives information about the proportion of variance in the dependent variables explained by the independent variables. Let $Y = \{y_i\}_{i=1}^n$ and $\hat{Y} = \{\hat{y}_i\}_{i=1}^n$ be the sets of target and predicted values, respectively. Let $\overline{y} = \frac{1}{n}\sum_{i=1}^n y_i$ be the mean of the target values. Then, the proportion of variance of sets Y and $\hat{Y}$ can be calculated with

$$R^2(Y,\hat{Y}) = 1 - \frac{RSS}{TSS}, \quad RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \quad TSS = \sum_{i=1}^{n}(y_i - \overline{y})^2, \tag{6.4}$$

where RSS is the sum of squares of differences between the target and predicted values and TSS is the total sum of squares. The R^2 score ranges typically between 0 and 1, where higher values correspond to better forecasts. When the score is negative, it indicates the forecast model yields values that are, on average, less accurate than simply using the $\overline{y}$ as the model predictor, making the model inappropriate for forecasting.

6.2.5 Experiment results

This section provides an overview of the models' forecasting performance for the stock close and daily volatility dynamics.

Stock close dynamics forecasting

In this section, we present the model's performance in forecasting the closing dynamics of stocks. Figure 6.4 shows the overall ranking of the models' performance, and Table 6.1 details their performance on individual stocks.

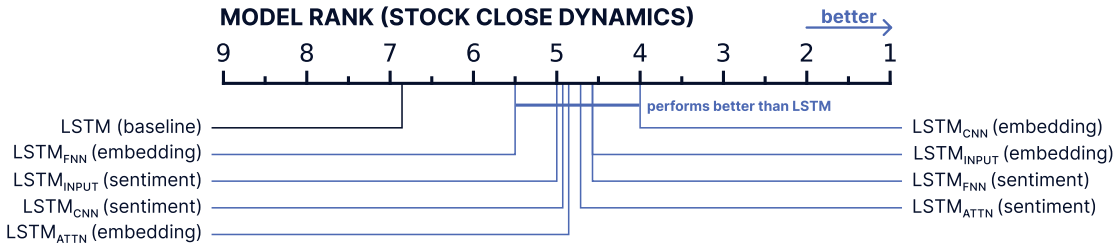


Figure 6.4: The models' performance ranks for forecasting stock close dynamics (based on R^2 score). All model extensions, on average, perform better than the baseline LSTM.

The results show that using the news stream as input enhances forecasting; on average, all eight model variants outperform the baseline LSTM in terms of their R^2 score. The highest-ranked model, LSTM_{CNN}, which uses sentence embeddings, outperforms other models on three stocks. Both $\text{LSTM}_{\text{ATTN}}$ model variants outperform others in four stocks, marking the best performance among the models. However, for two stocks (AAPL and WMT), none of the model extensions showed a statistically significant improvement over the baseline LSTM. Despite this, models that used sentence embeddings achieved the highest performance more often than those that used sentiment scores.

The stocks for which the models failed to improve performance significantly include AAPL, AMD, GOOG, GOOGL, TSLA, and WMT. After analyzing their corresponding

train and test sets, we found that they all have a higher average closing value in the test set than in the train set. While this might not directly explain the lack of performance improvement, it serves as a potential indicator of how the models might perform under such conditions.

Table 6.1: The models' performance of predicting the stock close dynamics (R^2 score reported). Reporting the absolute positive or negative difference denoted with the up (↑) and down (↓) arrow, respectively. Using symbols • and ∘, we highlight whether the mean absolute errors between the baseline LSTM and the comparing model are statistically significant (•) or not (∘) according to paired t-test at the significance level of $p = 0.05$.

	LSTM	$\text{LSTM}_{\text{INPUT}}$		LSTM_{FNN}		LSTM_{CNN}		$\text{LSTM}_{\text{ATTN}}$	
Stock	-	sentiment	embedding	sentiment	embedding	sentiment	embedding	sentiment	embedding
AAPL	0.9848	↑ 0.0015 ∘	↓ 0.0006 ∘	↓ 0.0510 •	↑ 0.0018 ∘	↓ 0.0482 •	↓ 0.0012 ∘	↓ 0.0087 •	↓ 0.0696 •
AMD	0.9873	↓ 0.0103 •	↓ 0.0002 ∘	↓ 0.0003 ∘	↓ 0.0760 •	↓ 0.0083 •	↑ 0.0002 ∘	↑ 0.0060 •	↑ 0.0034 •
AMZN	0.9807	↑ 0.0128 •	↑ 0.0066 •	↑ 0.0137 •	↑ 0.0127 •	↑ 0.0136 •	↑ 0.0134 •	↑ 0.0148 •	↑ 0.0132 •
GOOG	0.9848	↓ 0.0005 ∘	↑ 0.0032 ∘	↓ 0.0034 ∘	↓ 0.0010 ∘	↑ 0.0038 •	↑ 0.0008 ∘	↓ 0.0012 ∘	↑ 0.0079 •
GOOGL	0.9840	↑ 0.0051 •	↓ 0.0017 ∘	↑ 0.0049 •	↓ 0.0178 •	↓ 0.0025 ∘	↑ 0.0054 •	↓ 0.0367 •	↑ 0.0079 •
INTC	0.9888	↑ 0.0037 •	↑ 0.0018 ∘	↑ 0.0017 ∘	↑ 0.0036 •	↑ 0.0020 ∘	↑ 0.0029 ∘	↑ 0.0038 •	↑ 0.0032 •
MCD	0.9377	↑ 0.0533 •	↑ 0.0549 •	↑ 0.0549 •	↑ 0.0530 •	↑ 0.0548 •	↑ 0.0510 •	↓ 0.3352 •	↓ 0.0189 ∘
MSFT	0.8401	↑ 0.1443 •	↑ 0.1538 •	↑ 0.1501 •	↑ 0.1538 •	↑ 0.1070 •	↑ 0.1538 •	↑ 0.1468 •	↑ 0.1510 •
NFLX	0.9717	↑ 0.0129 •	↑ 0.0135 •	↑ 0.0186 •	↑ 0.0099 •	↑ 0.0137 •	↑ 0.0167 •	↑ 0.0232 •	↑ 0.0240 •
NKE	0.9790	↑ 0.0107 •	↑ 0.0094 •	↑ 0.0109 •	↑ 0.0107 •	↑ 0.0077 •	↑ 0.0096 •	↑ 0.0173 •	↑ 0.0081 •
NVDA	0.8813	↑ 0.1026 •	↓ 1.4030 •	↑ 0.1126 •	↑ 0.1124 •	↑ 0.0809 •	↑ 0.1129 •	↑ 0.1124 •	↑ 0.0591 •
PFE	0.9934	↑ 0.0019 •	↑ 0.0020 •	↑ 0.0014 ∘	↑ 0.0004 ∘	↑ 0.0024 •	↑ 0.0013 ∘	↑ 0.0021 •	↓ 0.0111 •
TSLA	0.9860	↑ 0.0007 ∘	↑ 0.0028 ∘	↑ 0.0040 •	↑ 0.0018 ∘	↑ 0.0033 ∘	↑ 0.0024 ∘	↑ 0.0001 ∘	↑ 0.0085 •
WMT	0.9822	↓ 0.0221 •	↑ 0.0009 ∘	↓ 0.0335 •	↓ 0.0349 •	↓ 0.0035 ∘	↓ 0.0077 •	↓ 0.0517 •	↓ 0.1813 •

Daily volatility dynamics forecasting

This section presents the models' performance in forecasting the daily volatility dynamics. Figure 6.5 displays the overall ranking of the models' performance, while Table 6.2 details their performance on individual stocks.

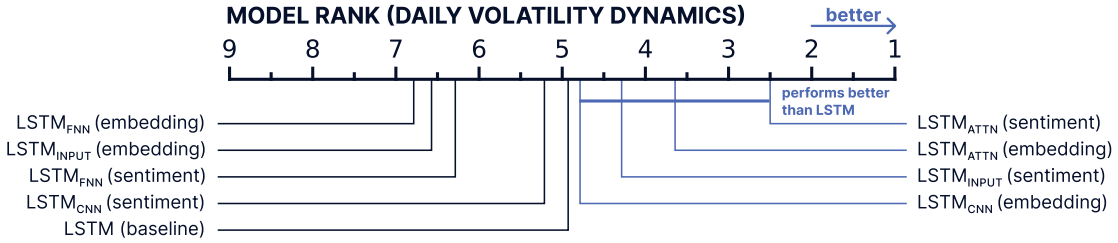


Figure 6.5: The models' performance ranks for forecasting daily volatility dynamics (based on R^2 score). On average, the $\text{LSTM}_{\text{ATTN}}$ models significantly outperform all other models.

The results show that only four model variants outperform the baseline LSTM. Out of these four, both of the $\text{LSTM}_{\text{ATTN}}$ models showcase the highest performance, where the variant using sentiment scores shows the most significant performance improvement of all the models. Unlike the market close dynamics, model extensions using sentiment scores outperform those using sentence embeddings. This suggests that sentiment scores of news events are more suitable for forecasting daily volatility than sentence embeddings. Generally, almost none of the performance changes are statistically significant. Moreover,

the overall performance scores of the models are lower than those for forecasting stock close dynamics, indicating that predicting daily volatility dynamics is a more challenging task.

Table 6.2: The models' performance of predicting the daily volatility dynamics (R^2 score reported). Reporting the absolute positive or negative difference denoted with the up (↑) and down (↓) arrow, respectively. Using symbols • and ∘, we highlight whether the mean absolute errors between the baseline LSTM and the comparing model are statistically significant (•) or not (∘) according to paired t-test at the significance level of $p = 0.05$.

	LSTM	$\text{LSTM}_{\text{INPUT}}$		LSTM_{FNN}		LSTM_{CNN}		$\text{LSTM}_{\text{ATTN}}$	
Stock	-	sentiment	embedding	sentiment	embedding	sentiment	embedding	sentiment	embedding
AAPL	0.7287	↑ 0.0101 ∘	↑ 0.0142 ∘	↓ 0.0022 ∘	↓ 0.0158 ∘	↑ 0.0095 ∘	↑ 0.0205 ∘	↑ 0.0231 ∘	↑ 0.0038 ∘
AMD	0.7777	↓ 0.0026 ∘	↓ 0.0116 ∘	↓ 0.0023 ∘	↓ 0.0053 ∘	↑ 0.0000 ∘	↓ 0.0054 ∘	↑ 0.0275 ∘	↑ 0.0224 ∘
AMZN	0.6776	↑ 0.0043 ∘	↓ 0.0082 ∘	↓ 0.0122 ∘	↓ 0.0574 ∘	↓ 0.0009 ∘	↓ 0.0144 ∘	↑ 0.0273 ∘	↓ 0.0095 ∘
GOOG	0.6323	↑ 0.0097 ∘	↓ 0.0212 ∘	↓ 0.0119 ∘	↓ 0.0296 ∘	↑ 0.0084 ∘	↓ 0.0152 ∘	↑ 0.0056 ∘	↑ 0.0228 ∘
GOOGL	0.6352	↑ 0.0069 ∘	↓ 0.0147 ∘	↓ 0.0105 ∘	↓ 0.0378 ∘	↑ 0.0099 ∘	↓ 0.0144 ∘	↑ 0.0108 ∘	↑ 0.0099 ∘
INTC	0.7843	↓ 0.0103 ∘	↑ 0.0001 ∘	↓ 0.0034 ∘	↓ 0.0032 ∘	↓ 0.0173 ∘	↑ 0.0005 ∘	↓ 0.0027 ∘	↓ 0.0037 ∘
MCD	0.6966	↑ 0.0406 ∘	↓ 0.0065 ∘	↑ 0.0228 ∘	↑ 0.0130 ∘	↓ 0.0030 ∘	↑ 0.0100 ∘	↑ 0.0567 ∘	↑ 0.0286 ∘
MSFT	0.7589	↑ 0.0107 ∘	↑ 0.0198 ∘	↑ 0.0100 ∘	↓ 0.0028 ∘	↑ 0.0085 ∘	↓ 0.0071 ∘	↑ 0.0150 ∘	↑ 0.0268 ∘
NFLX	0.7255	↑ 0.0047 ∘	↓ 0.0106 ∘	↑ 0.0002 ∘	↑ 0.0039 ∘	↓ 0.0002 ∘	↑ 0.0097 ∘	↑ 0.0224 ∘	↑ 0.0835 •
NKE	0.7086	↓ 0.0582 ∘	↓ 0.0391 ∘	↓ 0.0848 •	↓ 0.0308 ∘	↓ 0.0119 ∘	↓ 0.0099 ∘	↑ 0.0287 ∘	↓ 0.0056 ∘
NVDA	0.8369	↑ 0.0287 ∘	↑ 0.0270 ∘	↑ 0.0278 ∘	↑ 0.0185 ∘	↑ 0.0229 ∘	↑ 0.0313 ∘	↑ 0.0372 ∘	↑ 0.0287 ∘
PFE	0.8082	↓ 0.0075 ∘	↓ 0.0218 ∘	↓ 0.0100 ∘	↓ 0.0022 ∘	↓ 0.0009 ∘	↑ 0.0044 ∘	↓ 0.0247 ∘	↓ 0.0259 ∘
TSLA	0.8232	↓ 0.0007 ∘	↓ 0.0080 ∘	↓ 0.0049 ∘	↑ 0.0001 ∘	↓ 0.0010 ∘	↓ 0.0037 ∘	↑ 0.0185 ∘	↓ 0.0024 ∘
WMT	0.7773	↑ 0.0089 ∘	↓ 0.0201 ∘	↓ 0.0187 ∘	↓ 0.0688 ∘	↑ 0.0027 ∘	↑ 0.0120 ∘	↓ 0.0163 ∘	↑ 0.0147 ∘

6.3 Discussion

In this chapter, we explore the significance of streaming data fusion that supports multi-modal data sources. First, we provide the formal definition of streaming data fusion, focusing on heterogeneous data sources comprised mainly of numerical and categorical values. The definition includes what a data stream is and what functions can be used to aggregate, sample, and merge the data stream observations. Additionally, we introduce the forecast data stream and the feature vector. Next, we extend the formal definition to support multi-modal data sources, with a focus on integrating text streams for data stream forecasting. We then illustrate the practicality of this extension for predicting stock market dynamics, focusing on close predictions and daily volatility. We propose four methods of incorporating text stream observations, which are in the form of news events, into the forecasting models and compare them to the baseline LSTM model. We show the proposed approaches can significantly improve performance over the baseline model. The following discussion will focus on the findings concerning both the data fusion definition and the integration of text streams to support data stream forecasting.

6.3.1 Discussion on multi-modal data fusion

In this section, we discuss the multi-modal data fusion definition, including advantages and disadvantages, and theoretical and practical implications.

Improvements to the data fusion definition

While the definition was successfully used for modelling pipelines and forecasting in various domains, it could be improved in several ways. First, the data stream definition lacks functions like complex feature selections and advanced data stream resampler functions

(a comprehensive list of aggregate functions is available in [258], [259]). Furthermore, we presented two data stream mappers for text streams. Extending the definition to include other data stream mappers for text, including different data sources, such as video and audio streams, as well as discrete data points containing information relevant to the forecasting problem, would broaden the areas where the definition can be used to describe the data stream manipulation and preparation.

External model dependency of data stream mapper

The data stream mapper enables the transformation and mapping of observations within a data stream from one feature space to another. We demonstrate this with text streams, where text can be mapped into the sentiment score space using the sentiment analysis mapper, or into the sentence embedding space using the sentence embedding mapper. In both cases, the mapper simply describes how we transform the data stream values. However, in practice, a dedicated model is required to perform this operation. Since there are many approaches and models that extract sentiment scores and generate sentence embeddings, the data stream mapper, in practice, must always include the reference to the actual model used for the transformation. This is crucial for making experiments reproducible for other researchers.

6.3.2 Discussion on including text into time series forecasting models

This section focuses on including text in time series forecasting models. We discuss the models' performance and the proposed text inclusion methods, focusing on their advantages and disadvantages.

Models' performance

The proposed stock market forecasting models were evaluated on their ability to predict stock close prices and daily volatility dynamics. While all LSTM extensions outperformed the baseline in predicting future stock close prices, only half were more effective in forecasting daily volatility. The results show that considering both stock market dynamics the $\text{LSTM}_{\text{ATTN}}$ model, which uses sentiment scores, performed best, with an average rank of about 4.5 for stock close and 2.5 for daily volatility dynamics. This makes it a strong benchmark for future models. Additionally, this model uses the Attention mechanism to focus on particular values in the stock data stream that are more significant, according to the sentiment score of news events.

The model with the lowest overall performance is LSTM_{FNN} which uses sentence embeddings. This indicates that the linear Feed-Forward layer is not suitable for processing multi-dimensional vectors because it only transforms the embeddings linearly. The experimental results suggest that more complex operations, such as the use of a CNN or an Attention mechanism, are necessary to effectively handle these types of data.

Text stream inclusion methods

In our experiments, we incorporated the text stream using relatively simple components, such as a linear Feed-Forward layer, a shallow CNN, and a single Attention layer. These components demonstrated the ability to process and extract valuable information from the text stream. However, more advanced approaches could potentially enhance the forecasting models even further, as they could potentially uncover valuable connections between the data and text streams, which could improve the model's forecasting performance. Additionally, we focused on scenarios where the text stream consists of continuous text streams,

such as news events. Including discrete text information, such as periodic reports, could also improve the forecasting performance. Although these discrete data points can be represented as data streams, their relevance may diminish over time due to ongoing changes. Because of that, integrating a decaying component to simulate the decreasing importance of these data points over time might be beneficial. Although the experiments used news events, the proposed approaches can be generalized to any temporally ordered texts. Furthermore, it can be applied also to other domains, including medicine-related forecasting, traffic prediction, and predictive maintenance.

Chapter 7

Conclusions

> I am glad you are here with me. Here at the end of all things.
>
> J.R.R. Tolkien

In this thesis, we investigate the use of machine learning to analyze sequential textual data. We have explored various methods incorporating sentence structure and/or the temporal aspects of text documents for sequence-related tasks. This chapter provides an overview of our research findings. We first highlight the scientific contributions in Section 7.1. We then discuss our research experiments, focusing on verifying the hypotheses defined at the beginning of this document in Section 7.2. Finally, based on our findings, we propose potential future research directions in Section 7.3.

7.1 Contributions to Science

In our research, we developed several methods that consider the sequential nature of text data. This section outlines the scientific contributions of this thesis. We begin by discussing how we include sentence structure in measuring semantic text similarity in Section 7.1.1. Next, we explore online text clustering using neural networks in Section 7.1.2. Lastly, we examine streaming multi-modal data fusion in Section 7.1.3.

7.1.1 Semantic text similarity using sentence structure

We develop two novel unsupervised methods for measuring semantic text similarity, LM-EMD and OPWScore, which focus on the meaning and order of individual words [SC-1] and are evaluated for tasks of information retrieval and machine translation [SC-5].

In Section 4.1, we propose the LM-EMD, a model used for cross-lingual information retrieval to measure the document's relevancy using all of the contextual information of words in both the query and document texts. The model-generated output also provides insight into why the document is relevant. The LM-EMD model is compared to other word embedding and multilingual language model approaches on two information retrieval data sets with different query structures – The Large-Scale CLIR data set contains queries that are whole sentences, while the CLIRMatrix data set has short queries. The experiments show that the LM-EMD model performs similarly to the best-performing model on high-resource languages on the Large-Scale CLIR data set but has difficulties with Slavic languages, as shown on the CLIRMatrix data set. Furthermore, the experiments also show that information retrieval models have difficulties retrieving texts for shorter queries. We

explain how to interpret the LM-EMD model's output and show its capabilities in an example with an English query and German documents. An analysis of the impact an objective loss function has on training and fine-tuning cross-lingual information retrieval models shows a significant increase in performance when using the pairwise ranking instead of the cross-entropy loss function. The code for the experiments is publicly available on Github [196].

Section 4.2 proposes a novel OPWScore metric for measuring text generation model performance. It assigns a score by considering the generated text's adequacy and fluency based on its associated gold reference. We compare OPWScore with various n-gram matching and embedding-based metrics on two machine translation data sets, WMT18 and WMT20. We perform two experiments: the adequacy experiments assess how well the metrics match the human-judgement scores. Here, we show that the OPWScore performance is lower than that of the leading unsupervised embedding-based metric but outperforms the n-gram matching approaches. The second experiment focuses on fluency, measuring how sensitive the metrics are to a sentence's word permutation. We develop a novel statistic for measuring the metric's sensitivity to fluency-related sentence modifications. With it, we show that OPWScore has the highest sensitivity to word order permutations among the unsupervised embedding-based metrics. Thus, the metric shows it is possible to combine both adequacy and fluency when assessing the quality of the generated text. The code for the experiments is publicly available on Github [214].

7.1.2 Online text clustering with neural networks

We propose a method for creating multilingual news data sets, which we used to develop OG2021, a multilingual news dataset focused on the 2021 Tokyo Olympics [SC-2]. Additionally, we developed a new online news clustering algorithm, WAC, which employs optimal transport for measuring the news articles' contextual and temporal similarity [SC-3]. The WAC algorithm was tested on two news clustering data sets, including OG2021 [SC-5].

In Section 5.1, we introduce a method for creating novel multilingual news clustering data sets that automate the collection and clustering of news articles. The process includes these steps: (1) data acquisition, which gathers news articles from multiple sources using Event Registry, a system for acquiring and aggregating news; (2) data cleanup and preparation, preparing the news articles for the next step; (3) automatic news clustering, using a multilingual news clustering algorithm optimized for precision while maintaining a reasonable recall score, resulting in clusters of similar news articles; and (4) manual evaluation and annotation, where human annotators review the generated clusters and make necessary changes to ensure the homogeneity of the news clusters. Using this method, we created the OG2021 data set, which contains about 11k multilingual news articles reporting on the 2021 Tokyo Olympics. These articles are in nine different languages from five language groups. They are grouped into 1,350 distinct clusters, where 28% of clusters contain articles in two or more languages. We compare the OG2021 with another data set, CDET, from Miranda et al. [114]. Although the two data sets share some common attributes, they differ in the distribution of languages and the co-occurrence of languages within clusters. Nonetheless, both data sets are useful for evaluating multilingual news clustering algorithms. The code for creating the data set is publicly available on Github [238], while the data set is available on a public data repository [221], [222].

In Section 5.2, we propose WAC, an online multilingual news clustering algorithm that uses different distance metrics for article clustering. The algorithm works in two stages: The first stage creates article clusters by measuring how similar the articles are to the clusters based on their temporal and contextual similarity. The second stage merges these clus-

ters using optimal transport, which assesses the similarity between the clusters' contextual and temporal distributions. We tested three different strategies for clustering articles in the first stage: monolingual clustering with named entity similarity scores ($\text{WAC}_{\text{MONO+NER}}$), monolingual clustering without named entity similarity scores (WAC_{MONO}), and multilingual clustering without entity similarity scores ($\text{WAC}_{\text{MULTI}}$). We evaluated WAC against unsupervised and classification-based algorithms on the CDET and OG2021 data sets. The CDET data set results show that $\text{WAC}_{\text{MULTI}}$ performs comparably to classification-based algorithms and better than distance-based ones. On the OG2021 data set, all three clustering strategies yielded similar success, with $\text{WAC}_{\text{MULTI}}$ outperforming the others. However, the overall performance on OG2021 was lower than on CDET due to differences in data distribution. An ablation study indicated that WAC achieves the best results without using entity similarity and by performing multilingual clustering in the first phase. Furthermore, the cluster merging phase is a crucial component of WAC, as it substantially enhances the quality of the constructed event clusters. The code for the experiments is publicly available on Github [264].

7.1.3 Streaming multi-modal data fusion

We propose a formal definition of data fusion, initially focusing on data sources related to the IoT domain. This definition was expanded to incorporate heterogeneous data stream sources, including text data. This extended definition then applied to predict stock market dynamics by combining news streams with stock market data, proposing four extensions to the LSTM model [SC-4]. The stock market dynamics models were evaluated and compared across various data sets related to these issues [SC-5].

Section 6.1 defines streaming data fusion of heterogeneous data sources within the Internet of Things domain. We introduce a new definition of a data stream, which includes time, and several data stream functions, including the aggregate, resampler, and merger, used to summarize, synchronize and merge data streams. Furthermore, we define a forecast data stream and how feature vectors are constructed using the (resampled) observations and aggregates. The definition was used to describe a streaming data fusion framework [247]. Since we focus on the definition of streaming data fusion, we do not include a detailed description of this framework; instead, we direct the reader to the cited paper for further details.

Section 6.2 expands the formal definition of data fusion to include multi-modal data sources, including text data. We define a data stream as observations that can be one-dimensional or multi-dimensional vectors. Then, a new data stream function called mapper, which maps individual observations from one feature space to another, is introduced. This function is particularly useful for converting data streams, such as news events, into formats compatible with machine learning models. Furthermore, two mapper functions were introduced: one that extracts sentiment scores and another that obtains sentence embeddings from text. Following mapping, the data stream can be processed using standard aggregate, resampler, and merger functions. Applying this extended framework to stock market forecasting, four methods to integrate text data with time series forecasting models are proposed: straightforward inclusion of text features and utilizing Feed-Forward, CNN, and Attention layers to process and integrate these with data stream inputs. We collected stock market data and related news from the Event Registry for our predictions of future stock close prices and daily volatility dynamics. Our experiments demonstrate that incorporating text features enhances model performance, with the $\text{LSTM}_{\text{ATTN}}$ model showing the best overall performance on both problems. Additionally, the experiments found that forecasting daily volatility dynamics poses a greater challenge than predicting stock close dynamics, as reflected by lower R^2 scores in the models.

7.2 Discussion

This section evaluates whether the objectives of this thesis were met and supports our hypotheses with evidence. The **first goal** was to develop methods for measuring sentence similarity that incorporate word sequence information. We introduced two methods, LM-EMD and OPWScore, which utilize optimal transport to calculate the distance between word distributions in sentences. Specifically, OPWScore adds a regularization element to the standard optimal transport method, limiting mappings to similarly indexed words, thereby factoring in the word sequence. By doing so, we believe we have achieved the set goal. Experimentally, we demonstrate that our methods yield interpretable results. The experiments with OPWScore reveal that incorporating word sequence into text similarity assessments does make the method more sensitive to changes in fluency. However, although it does not surpass the top-performing embedding-based method, it still outperforms traditional n-gram matching algorithms. Although enforcing word sequence mapping does not enhance performance significantly, it introduces an additional layer of analysis absent in other text similarity measures. Due to these findings, we cannot conclusively confirm or reject **Hypothesis 1**, which suggests that text similarity measures that include word sequences generate more contextually aware scores. While the inclusion of word sequence does improve the measures from the fluency perspective, it does not consistently perform better in experiments focused on adequacy. Thus, the hypothesis stands partially supported, recognizing the benefits in one aspect of text similarity assessment while acknowledging room for improvement in others.

The **second goal** was to develop a novel news clustering approach that uses unconventional similarity methods. We achieve this goal with the WAC algorithm, which uses optimal transport to measure clusters' contextual and temporal similarity. Including these unconventional similarity measures enables WAC to outperform the unsupervised clustering algorithms and is comparable to the best classifier-based algorithms. Furthermore, we evaluate WAC on two data sets, showing that it performs comparably or even better than other algorithms. This confirms **Hypothesis 2**, stating that using structure including similarity metrics in news stream clustering algorithms improves the quality and homogeneity of the generated news clusters.

The **third goal** was to develop methods for combining text and data streams to improve data stream predictions. We defined multi-modal data fusion of heterogeneous data sources, including mapping, aggregating, sampling and merging different data streams. Using this definition, we developed four methods for including text streams into time series forecasting models, specifically for predicting stock market dynamics. The results show that careful preparation of the data streams and components for including text streams can improve the forecasting model's performance, thus achieving our goal. Furthermore, we partially confirm **Hypothesis 3**, which includes relevant textual information in numeric data streams, improves the performance of data stream predictions. Our experiments utilized news events as external inputs to inform stock market dynamics. We also implemented specific components to include the textual information in the LSTM models. The positive outcomes from these experiments support the hypothesis, though further testing across additional forecasting models would provide a more exhaustive evaluation of the hypothesis's validity.

Finally, all of the proposed approaches have been evaluated on data sets appropriate for the given problem we were trying to solve, thus achieving the **fourth goal**, which is evaluating the proposed methods on various data sets.

7.3 Further Work

This section organizes further work into three parts, each related to one of the main chapters of this thesis.

Semantic text similarity using sentence structure. Regarding the experiments of the LM-EMD and OPWscore methods, there are some possible research directions. First, making the models smaller by using model distillation and pruning would potentially speed up the calculation of the relevance scores. Moreover, integrating different text features, such as part-of-speech tags and syntactic embedding [265], into optimal transport calculation could potentially improve the model's performance, as it would include information about the sentence structure into consideration.

Several strategies could be explored to improve outcomes when using optimal transport for measuring sentence similarity. Calculating the p-Wasserstein distance for $p \geq 2$ and experimenting with different prior distributions in the order-preserving optimal transport problem are promising options to refine the similarity measures. Moreover, leveraging outputs from different layers of various language models could enrich the assessment metrics, as these layers capture different aspects of linguistic information [103].

Furthermore, recent developments in large language models (LLMs) also offer exciting research opportunities. These models exhibit impressive generative capabilities, which are a product of how the models generate word/token representations. Exploring the potential of utilizing these representations in measuring text similarity could be a promising avenue for future research. Additionally, the LLMs can open up possibilities for unconventional approaches to measuring text similarity that have not yet been exposed.

Online text clustering with neural networks. The method we proposed for creating news clustering data sets offers a versatile approach to generating novel data collections that can be used to train and evaluate clustering algorithms. The method can be extended beyond news articles and adapted for temporally ordered text streams, such as social media posts, blog entries, or academic articles. It can be utilized to create novel data collections that can then be used for training and evaluating various clustering algorithms. By allowing for the generation of temporal text collections that include multiple labels, this approach broadens the scope of applications for different algorithms.

Considering the results of the WAC clustering algorithm, some possible research directions exist. First, the algorithm can be enhanced by implementing a loose variation of cluster merging, where only a sample of articles from each cluster would be considered when assessing when to merge the clusters. Furthermore, while the hyper-parameters were manually set in our experiments, using methods to learn the optimal solutions could potentially improve the quality of the clusters.

Similar to the previous subsection, using LLMs could potentially improve both the data set creation process and the news clustering algorithm. They could be used to extract the relevant entities from the text and form them into the format that would be optimal for the task at hand; in the case of news articles, they could be fine-tuned to extract the answers to the journalist questions WHO, WHAT, WHERE, WHEN, which would then be used as the input information to the clustering algorithm. Furthermore, the LLMs could summarize the news clusters and thus provide us with valuable information regarding world events.

Streaming multi-modal data fusion. The proposed definition of multi-modal data fusion was showcased on data and text streams. A possible direction for research is to include other data modalities, including video streams, which include visual information that can help forecast traffic in real-time. Furthermore, the development of functions

for mapping, aggregating, sampling and merging multi-modal data streams would provide additional support in combining different sources of information for time series forecasting.

To evaluate the performance of incorporating text in time series forecasting models versus the baseline model, we compared the baseline LSTM with its extensions. Although this proved effective as a proof-of-concept, other models could also be considered. These range from traditional models such as ridge regression, k-nearest neighbors, decision trees, gradient boosting regression, and random forests to more recent neural network approaches like the transformer-based forecasting model. The latter is particularly interesting in the research community, which continues investigating the best ways to utilize the Attention mechanism for time series forecasting.

References

[1] S. Robertson, "Understanding inverse document frequency: On theoretical arguments for IDF," *The Journal of documentation; devoted to the recording, organization and dissemination of specialized knowledge*, vol. 60, pp. 503–520, 2004. DOI: 10.1108/00220410410560582.

[2] W. Uther, D. Mladenić, M. Ciaramita, *et al.*, "TF–IDF," in *Encyclopedia of Machine Learning*, Springer US, 2011, pp. 986–987. DOI: 10.1007/978-0-387-30164-8_832.

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv [cs.CL]*, 2013.

[4] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162.

[5] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017. DOI: 10.1162/tacl_a_00051.

[6] C. Xing, D. Wang, C. Liu, and Y. Lin, "Normalized word embedding and orthogonal transform for bilingual word translation," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, 2015. DOI: 10.3115/v1/n15-1104.

[7] K. Huang, M. Gardner, E. Papalexakis, *et al.*, "Translation invariant word embeddings," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2015. DOI: 10.18653/v1/d15-1127.

[8] A. Conneau, G. Lample, M. Ranzato, L. Denoyer, and H. Jégou, "Word translation without parallel data," *arXiv [cs.CL]*, 2017.

[9] A. Joulin, P. Bojanowski, T. Mikolov, H. Jégou, and E. Grave, "Loss in translation: Learning bilingual word mapping with a retrieval criterion," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2018. DOI: 10.18653/v1/d18-1330.

[10] G. Glavaš, M. Franco-Salvador, S. P. Ponzetto, and P. Rosso, "A resource-light method for cross-lingual semantic textual similarity," *Knowledge-based systems*, vol. 143, pp. 1–9, 2018. DOI: 10.1016/j.knosys.2017.11.041.

[11] S. Arora, Y. Liang, and T. Ma, "A simple but though-to-beat baseline for sentence embedding," in *Proceedings of the International Conference on Learning Representations*, 2017.

[12] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International Conference on Machine Learning*, PMLR, 2014, pp. 1188–1196.

[13] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur, "Recurrent neural network based language model," *Interspeech*, 2010.

[14] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Curran Associates Inc., 2017, pp. 6000–6010.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

[16] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, ser. 517, 2019, pp. 5753–5763.

[17] Y. Liu, M. Ott, N. Goyal, *et al.*, "RoBERTa: A robustly optimized BERT pretraining approach," *arXiv [cs.CL]*, 2019.

[18] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," in *International Conference on Learning Representations*, 2019.

[19] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter," *arXiv [cs.CL]*, 2019.

[20] T. Pires, E. Schlinger, and D. Garrette, "How multilingual is multilingual BERT?" In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2019, pp. 4996–5001.

[21] A. Conneau and G. Lample, "Cross-lingual language model pretraining," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[22] A. Conneau, K. Khandelwal, N. Goyal, *et al.*, "Unsupervised cross-lingual representation learning at scale," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2020, pp. 8440–8451. DOI: 10.18653/v1/2020.acl-main.747.

[23] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[24] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, *Language models are unsupervised multitask learners*, 2019.

[25] T. B. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, ser. 33, 2020, pp. 1877–1901.

[26] M. Lewis, Y. Liu, N. Goyal, *et al.*, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2020, pp. 7871–7880.

[27] C. Raffel, N. Shazeer, A. Roberts, *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of machine learning research: JMLR*, vol. 21, pp. 5485–5551, 2020.

[28] R. Taylor, M. Kardas, G. Cucurull, *et al.*, "Galactica: A large language model for science," *arXiv [cs.CL]*, 2022.

[29] L. Ouyang, J. Wu, X. Jiang, *et al.*, "Training language models to follow instructions with human feedback," *arXiv [cs.CL]*, 2022.

[30] P. F. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, Curran Associates Inc., 2017, pp. 4302–4310. DOI: 10.5555/3294996.3295184.

[31] N. Stiennon, L. Ouyang, J. Wu, *et al.*, "Learning to summarize with human feedback," *Advances in Neural Information Processing Systems*, vol. 33, pp. 3008–3021, 2020.

[32] H. Touvron, T. Lavril, G. Izacard, *et al.*, "LLaMA: Open and efficient foundation language models," *arXiv [cs.CL]*, 2023.

[33] H. Touvron, L. Martin, K. Stone, *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv [cs.CL]*, 2023.

[34] E. Almazrouei, H. Alobeidli, A. Alshamsi, *et al.*, "The falcon series of open language models," *arXiv [cs.CL]*, 2023.

[35] A. Q. Jiang, A. Sablayrolles, A. Mensch, *et al.*, "Mistral 7B," *arXiv [cs.CL]*, 2023.

[36] R. Anil, A. M. Dai, O. Firat, *et al.*, "PaLM 2 technical report," *arXiv [cs.CL]*, 2023.

[37] S. Wu, O. Irsoy, S. Lu, *et al.*, "BloombergGPT: A large language model for finance," *arXiv [cs.LG]*, 2023.

[38] H. Yang, X.-Y. Liu, and C. D. Wang, "FinGPT: Open-source financial large language models," *arXiv [q-fin.ST]*, 2023.

[39] R. Li, L. B. Allal, Y. Zi, *et al.*, "StarCoder: May the source be with you!" *arXiv [cs.CL]*, 2023.

[40] G. Wang, G. Yang, Z. Du, L. Fan, and X. Li, "ClinicalGPT: Large language models finetuned with diverse medical data and comprehensive evaluation," *arXiv [cs.CL]*, 2023.

[41] M. Moor, Q. Huang, S. Wu, *et al.*, "Med-flamingo: A multimodal medical few-shot learner," *arXiv [cs.CV]*, 2023.

[42] Z. Chen, A. H. Cano, A. Romanou, *et al.*, "MEDITRON-70B: Scaling medical pretraining for large language models," *arXiv [cs.CL]*, 2023.

[43] C. Wu, X. Zhang, Y. Zhang, Y. Wang, and W. Xie, "PMC-LLaMA: Further finetuning LLaMA on medical papers," *arXiv [cs.CL]*, 2023.

[44] M. Kosinski, "Theory of mind might have spontaneously emerged in large language models," *arXiv [cs.CL]*, 2023.

[45] S. Bubeck, V. Chandrasekaran, R. Eldan, *et al.*, "Sparks of artificial general intelligence: Early experiments with GPT-4," *arXiv [cs.CL]*, 2023.

[46] A. Moradi Dakhel, V. Majdinasab, A. Nikanjam, F. Khomh, M. C. Desmarais, and Z. M. (Jiang, "GitHub copilot AI pair programmer: Asset or liability?" *The Journal of systems and software*, vol. 203, p. 111 734, 2023. DOI: 10.1016/j.jss.2023.111734.

[47] R. Schaeffer, B. Miranda, and S. Koyejo, "Are emergent abilities of large language models a mirage?" In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[48] B. Romera-Paredes, M. Barekatain, A. Novikov, *et al.*, "Mathematical discoveries from program search with large language models," *Nature*, pp. 1–3, 2023. DOI: 10.1038/s41586-023-06924-6.

[49] N. Zhang, Y. Yao, B. Tian, *et al.*, "A comprehensive study of knowledge editing for large language models," *arXiv [cs.CL]*, 2024.

[50] S. Biderman, H. Schoelkopf, Q. Anthony, *et al.*, "Pythia: A suite for analyzing large language models across training and scaling," *arXiv [cs.CL]*, 2023.

[51] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv [cs.LG]*, 2023.

[52] A. Bertsch, U. Alon, G. Neubig, and M. R. Gormley, "Unlimiformer: Long-range transformers with unlimited length input," *arXiv [cs.CL]*, 2023.

[53] B. Peng, E. Alcaide, Q. Anthony, *et al.*, "RWKV: Reinventing RNNs for the transformer era," *arXiv [cs.CL]*, 2023.

[54] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient finetuning of quantized LLMs," *arXiv [cs.LG]*, 2023.

[55] H. Chen, F. Jiao, X. Li, *et al.*, "ChatGPT's one-year anniversary: Are open-source large language models catching up?" *arXiv [cs.CL]*, 2023.

[56] C. Villani, *Optimal Transport*, 2009th ed., ser. Grundlehren der mathematischen Wissenschaften. Springer, 2008.

[57] R. Saabni, "Efficient word image retrieval using earth movers distance embedded to wavelets coefficients domain," in *2013 12th International Conference on Document Analysis and Recognition*, IEEE, 2013. DOI: 10.1109/icdar.2013.70.

[58] G. Yu, X. Xu, Z. Yan, and H. Yu, "Accelerating earth movers distance with instruction set extension for image retrieval," in *2018 China Semiconductor Technology International Conference (CSTIC)*, IEEE, 2018. DOI: 10.1109/cstic.2018.8369339.

[59] C. Zhang, Y. Cai, G. Lin, and C. Shen, "DeepEMD: Few-shot image classification with differentiable earth mover's distance and structured classifiers," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2020. DOI: 10.1109/cvpr42600.2020.01222.

[60] J. Li, X. Liu, H. Zhao, R. Xu, M. Yang, and Y. Jin, "BERT-EMD: Many-to-many layer mapping for BERT compression with earth mover's distance," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, 2020. DOI: 10.18653/v1/2020.emnlp-main.242.

[61] S. Kumar, S. Chakrabarti, and S. Roy, "Earth mover's distance pooling over siamese LSTMs for automatic short answer grading," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, International Joint Conferences on Artificial Intelligence Organization, 2017, pp. 2046–2052. DOI: 10.24963/ijcai.2017/284.

[62] L. Wu, I. E.-H. Yen, K. Xu, *et al.*, "Word mover's embedding: From Word2Vec to document embedding," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2018. DOI: 10.18653/v1/d18-1482.

[63] M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transport," *Advances in neural information processing systems*, vol. 26, 2013.

[64] J. A. Nasir, I. Varlamis, and S. Ishfaq, "A knowledge-based semantic framework for query expansion," *Information processing & management*, vol. 56, pp. 1605–1617, 2019. DOI: 10.1016/j.ipm.2019.04.007.

[65] F. Ensan and F. Al-Obeidat, "Relevance-based entity selection for ad hoc retrieval," *Information processing & management*, vol. 56, pp. 1645–1666, 2019. DOI: 10.1016/j.ipm.2019.05.005.

[66] S. Jain, K. R. Seeja, and R. Jindal, "A fuzzy ontology framework in information retrieval using semantic query expansion," *International Journal of Information Management Data Insights*, vol. 1, p. 100 009, 2021. DOI: 10.1016/j.jjimei.2021.100009.

[67] S. Kuzi, A. Shtok, and O. Kurland, "Query expansion using word embeddings," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, ACM, 2016. DOI: 10.1145/2983323.2983876.

[68] H. K. Azad and A. Deepak, "Query expansion techniques for information retrieval: A survey," *Information processing & management*, vol. 56, pp. 1698–1735, 2019. DOI: 10.1016/j.ipm.2019.05.009.

[69] R. Gao and C. Shah, "Toward creating a fairer ranking in search engine results," *Information processing & management*, vol. 57, p. 102 138, 2020. DOI: 10.1016/j.ipm.2019.102138.

[70] S. Marchesin, A. Purpura, and G. Silvello, "Focal elements of neural information retrieval models. an outlook through a reproducibility study," *Information processing & management*, vol. 57, p. 102 109, 2020. DOI: 10.1016/j.ipm.2019.102109.

[71] F. Ture and E. Boschee, "Learning to translate: A query-specific combination approach for cross-lingual information retrieval," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, 2014. DOI: 10.3115/v1/d14-1064.

[72] M. Artetxe, G. Labaka, and E. Agirre, "Learning principled bilingual mappings of word embeddings while preserving monolingual invariance," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2016, pp. 2289–2294. DOI: 10.18653/v1/d16-1250.

[73] S. L. Smith, D. H. P. Turban, S. Hamblin, and N. Y. Hammerla, "Offline bilingual word vectors, orthogonal transformations and the inverted softmax," *arXiv [cs.CL]*, 2017.

[74] T. Brychcín, "Linear transformations for cross-lingual semantic textual similarity," *Knowledge-based systems*, vol. 187, p. 104 819, 2020. DOI: 10.1016/j.knosys.2019.06.027.

[75] R. Litschko, G. Glavaš, S. P. Ponzetto, and I. Vulić, "Unsupervised cross-lingual information retrieval using monolingual data only," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, ACM, 2018. DOI: 10.1145/3209978.3210157.

[76] W. Yang, H. Zhang, and J. Lin, "Simple applications of BERT for ad hoc document retrieval," *arXiv [cs.IR]*, 2019.

[77] S. MacAvaney, A. Yates, A. Cohan, and N. Goharian, "CEDR: Contextualized embeddings for document ranking," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, 2019. DOI: 10.1145/3331184.3331317.

[78] R. Nogueira and K. Cho, "Passage re-ranking with BERT," *arXiv [cs.IR]*, 2019.

[79] Z. Akkalyoncu Yilmaz, S. Wang, W. Yang, H. Zhang, and J. Lin, "Applying BERT to document retrieval with birch," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, Association for Computational Linguistics, 2019. DOI: 10.18653/v1/d19-3004.

[80] Y. Qiao, C. Xiong, Z. Liu, and Z. Liu, "Understanding the behaviors of BERT in ranking," *arXiv [cs.IR]*, 2019.

[81] W. Guo, X. Liu, S. Wang, *et al.*, "DeText: A deep text ranking framework with BERT," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, ACM, 2020. DOI: 10.1145/3340531.3412699.

[82] S. Han, X. Wang, M. Bendersky, and M. Najork, "Learning-to-rank with BERT in TF-ranking," *arXiv [cs.IR]*, 2020.

[83] Z. Jiang, A. El-Jaroudi, W. Hartmann, D. Karakos, and L. Zhao, "Cross-lingual information retrieval with BERT," in *Proceedings of the Workshop on Cross-Language Search and Summarization of Text and Speech (CLSSTS2020)*, European Language Resources Association, 2020, pp. 26–31.

[84] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using siamese BERT-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Association for Computational Linguistics, 2019. DOI: 10.18653/v1/d19-1410.

[85] N. Reimers and I. Gurevych, "Making monolingual sentence embeddings multilingual using knowledge distillation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, 2020, pp. 4512–4525. DOI: 10.18653/v1/2020.emnlp-main.365.

[86] S. Harrat, K. Meftouh, and K. Smaili, "Machine translation for arabic dialects (survey)," *Information processing & management*, vol. 56, pp. 262–273, 2019. DOI: 10.1016/j.ipm.2017.08.003.

[87] W. Farhan, B. Talafha, A. Abuammar, *et al.*, "Unsupervised dialectal neural machine translation," *Information processing & management*, vol. 57, p. 102 181, 2020. DOI: 10.1016/j.ipm.2019.102181.

[88] F. Zaman, M. Shardlow, S.-U. Hassan, N. R. Aljohani, and R. Nawaz, "HTSS: A novel hybrid text summarisation and simplification architecture," *Information processing & management*, vol. 57, p. 102 351, 2020. DOI: 10.1016/j.ipm.2020.102351.

[89] B. Mutlu, E. A. Sezer, and M. A. Akcayol, "Candidate sentence selection for extractive text summarization," *Information processing & management*, vol. 57, p. 102 359, 2020. DOI: 10.1016/j.ipm.2020.102359.

[90] A. Nawaz, M. Bakhtyar, J. Baber, I. Ullah, W. Noor, and A. Basit, "Extractive text summarization models for urdu language," *Information processing & management*, vol. 57, p. 102 383, 2020. DOI: 10.1016/j.ipm.2020.102383.

[91] A. Alomari, N. Idris, A. Q. Sabri, and I. Alsmadi, "Deep reinforcement and transfer learning for abstractive text summarization: A review," *Computer speech & language*, vol. 71, p. 101 276, 2022. DOI: 10.1016/j.csl.2021.101276.

[92] Q. Zhao, J. Niu, X. Liu, W. He, and S. Tang, "Generation of coherent multi-sentence texts with a coherence mechanism," *Computer speech & language*, vol. 78, p. 101 457, 2023. DOI: 10.1016/j.csl.2022.101457.

[93] M. Bani-Almarjeh and M.-B. Kurdy, "Arabic abstractive text summarization using RNN-based and transformer-based architectures," *Information processing & management*, vol. 60, p. 103 227, 2023. DOI: 10.1016/j.ipm.2022.103227.

[94] E. Seifossadat and H. Sameti, "Stochastic data-to-text generation using syntactic dependency information," *Computer speech & language*, vol. 76, p. 101 388, 2022. DOI: 10.1016/j.csl.2022.101388.

[95] S. Demir and S. Oktem, "A benchmark dataset for turkish data-to-text generation," *Computer speech & language*, vol. 77, p. 101 433, 2023. DOI: 10.1016/j.csl.2022.101433.

[96] M. Guan, S. K. Mondal, H.-N. Dai, and H. Bao, "Reinforcement learning-driven deep question generation with rich semantics," *Information processing & management*, vol. 60, p. 103 232, 2023. DOI: 10.1016/j.ipm.2022.103232.

[97] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2002, pp. 311–318. DOI: 10.3115/1073083.1073135.

[98] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*, Association for Computational Linguistics, 2004, pp. 74–81.

[99] S. Banerjee and A. Lavie, "METEOR: An automatic metric for MT evaluation with improved correlation with human judgments," in *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, Association for Computational Linguistics, 2005, pp. 65–72.

[100] R. Vedantam, C. Lawrence Zitnick, and D. Parikh, "CIDEr: Consensus-based image description evaluation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 4566–4575.

[101] T. Sellam, D. Das, and A. Parikh, "BLEURT: Learning robust metrics for text generation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2020. DOI: 10.18653/v1/2020.acl-main.704.

[102] W. Zhao, M. Peyrard, F. Liu, Y. Gao, C. M. Meyer, and S. Eger, "MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Association for Computational Linguistics, 2019. DOI: 10.18653/v1/d19-1053.

[103] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "BERTScore: Evaluating text generation with BERT," in *Eighth International Conference on Learning Representations*, 2020.

[104] R. Rei, C. Stewart, A. C. Farinha, and A. Lavie, "COMET: A neutral framework for MT evaluation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2020, pp. 2685–2702.

[105] W. Yuan, G. Neubig, and P. Liu, "BARTScore: Evaluating generated text as text generation," in *Advances in Neural Information Processing Systems*, M. Ranzato and A. Beygelzimer and Y. Dauphin and P.S. Liang and J. Wortman Vaughan, Ed., Curran Associates, Inc., 2021, pp. 27 263–27 277.

[106] A. B. Sai, T. Dixit, D. Y. Sheth, S. Mohan, and M. M. Khapra, "Perturbation CheckLists for evaluating NLG evaluation metrics," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2021. DOI: 10.18653/v1/2021.emnlp-main.575.

[107] J. Lin, Y. Wang, M. Efron, and G. Sherman, "Overview of the TREC-2014 microblog track," in *Text Retrieval Conference*, 2014.

[108] J. Lin, M. Efron, Y. Wang, G. Sherman, and E. Voorhees, "Overview of the TREC-2015 microblog track," in *Text Retrieval Conference*, 2015.

[109] A. Gulli, *AG news*, http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html, (accessed 8 August 2024), 2004.

[110] K. Lang, "NewsWeeder: Learning to filter netnews," in *Machine Learning Proceedings 1995*, Elsevier, 1995, pp. 331–339. DOI: 10.1016/b978-1-55860-377-6.50048-7.

[111] D. Lewis, *Reuters-21578 text categorization collection*, 1997. DOI: 10.24432/C52G6M.

[112] C. Wayne, G. R. Doddington, J. G. Fiscus, *et al.*, *TDT2 multilanguage text version 4.0*, 2001. DOI: 10.35111/ZFJ3-TP72.

[113] G. Leban, B. Fortuna, J. Brank, and M. Grobelnik, "Event registry: Learning about world events from news," in *Proceedings of the 23rd International Conference on World Wide Web*, ACM, 2014. DOI: 10.1145/2567948.2577024.

[114] S. Miranda, A. Znotiņš, S. B. Cohen, and G. Barzdins, "Multilingual clustering of streaming news," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2018, pp. 4535–4544. DOI: 10.18653/v1/d18-1483.

[115] K. Leetaru and P. A. Schrodt, "Gdelt: Global data on events, location, and tone, 1979–2012," *ISA annual convention*, 2013.

[116] H. Kwak and J. An, "Two tales of the world: Comparison of widely used world news datasets GDELT and EventRegistry," in *Proceedings of the International AAAI Conference on Web and Social Media*, 2016. DOI: 10.1609/icwsm.v10i1.14763.

[117] E. Maden and P. Karagoz, "Recent methods on short text stream clustering: A survey study," *Wiley interdisciplinary reviews. Computational statistics*, 2023. DOI: 10.1002/wics.1610.

[118] J. Kumar, S. U. Din, Q. Yang, R. Kumar, and J. Shao, "An online semantic-enhanced graphical model for evolving short text stream clustering," *IEEE transactions on cybernetics*, vol. 52, pp. 13 809–13 820, 2022. DOI: 10.1109/TCYB.2021.3108897.

[119] J. Yin, D. Chao, Z. Liu, W. Zhang, X. Yu, and J. Wang, "Model-based clustering of short text streams," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, 2018. DOI: 10.1145/3219819.3220094.

[120] J. Qiang, W. Xu, Y. Li, Y. Yuan, and Y. Zhu, "Lifelong learning augmented short text stream clustering method," *IEEE access: practical innovations, open solutions*, vol. 9, pp. 70 493–70 501, 2021. DOI: 10.1109/access.2021.3078096.

[121] X. Si, P. Li, X. Hu, and Y. Zhang, "An online dirichlet model based on sentence embedding and DBSCAN for noisy short text stream clustering," in *2022 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2022, pp. 01–08. DOI: 10.1109/ijcnn55064.2022.9892414.

[122] M. R. H. Rakib and M. Asaduzzaman, "Fast clustering of short text streams using efficient cluster indexing and dynamic similarity thresholds," *arXiv [cs.IR]*, 2021.

[123] A. Najafi, A. Gholipour-Shilabin, R. Dehkharghani, A. Mohammadpur-Fard, and M. Asgari-Chenaghlu, "ComStreamClust: A communicative multi-agent approach to text clustering in streaming data," *Annals of data science*, 2022. DOI: 10.1007/s40745-022-00426-4.

[124] W. Zhang, C. Dong, J. Yin, and J. Wang, "Attentive representation learning with adversarial training for short text clustering," *IEEE transactions on knowledge and data engineering*, vol. 34, pp. 5196–5210, 2022. DOI: 10.1109/tkde.2021.3052244.

[125] S. Yang, G. Huang, X. Zhou, and Y. Xiang, "Dynamic clustering of stream short documents using evolutionary word relation network," in *Communications in Computer and Information Science*, ser. Communications in computer and information science, Springer Singapore, 2020, pp. 418–428. DOI: 10.1007/978-981-15-2810-1_40.

[126] S. Yang, G. Huang, X. Zhou, V. Mak, and J. Yearwood, "EWNStream+: Effective and real-time clustering of short text streams using evolutionary word relation network," *International journal of information technology & decision making*, vol. 20, pp. 341–370, 2021. DOI: 10.1142/s0219622021500024.

[127] C. C. Aggarwal and P. S. Yu, "A framework for clustering massive text and categorical data streams," in *Proceedings of the 2006 SIAM International Conference on Data Mining*, Society for Industrial and Applied Mathematics, 2006. DOI: 10.1137/1.9781611972764.44.

[128] C. C. Aggarwal and P. S. Yu, "On clustering massive text and categorical data streams," *Knowledge and information systems*, vol. 24, pp. 171–196, 2010. DOI: 10.1007/s10115-009-0241-z.

[129] J. Brank, G. Leban, and M. Grobelnik, "Annotating documents with relevant wikipedia concepts," in *Proceedings of Slovenian KDD Conference on Data Mining and Data Warehouses (SiKDD)*, 2017.

[130] J. Brank, G. Leban, and M. Grobelnik, "Semantic annotation of documents based on wikipedia concepts," *Informatica. An International Journal of Computing and Informatics*, vol. 42, 2018.

[131] X. Qu, J. Yang, B. Wu, and H. Xin, "A news event detection algorithm based on key elements recognition," in *2016 IEEE First International Conference on Data Science in Cyberspace (DSC)*, IEEE, 2016. DOI: 10.1109/dsc.2016.53.

[132] P. Laban and M. A. Hearst, "newsLens: Building and visualizing long-ranging news stories," in *Proceedings of the Events and Stories in the News Workshop*, 2017, pp. 1–9. DOI: 10.18653/v1/W17-2701.

[133] F. K. Örs, S. Yeniterzi, and R. Yeniterzi, "Event clustering within news articles," in *Proceedings of the Workshop on Automated Extraction of Socio-political Events from News 2020*, 2020, pp. 63–68.

[134] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, ACM, 1992. DOI: 10.1145/130385.130401.

[135] N. Cristianini and E. Ricci, "Support vector machines: 1992; boser, guyon, vapnik," in *Encyclopedia of Algorithms*, Springer US, 2008, pp. 928–932. DOI: 10.1007/978-0-387-30162-4_415.

[136] J. Rupnik, A. Muhic, G. Leban, P. Skraba, B. Fortuna, and M. Grobelnik, "News across languages - cross-lingual document similarity and event tracking," *The journal of artificial intelligence research*, vol. 55, pp. 283–316, 2016. DOI: 10.1613/jair.4780.

[137] M. Linger and M. Hajaiej, "Batch clustering for multilingual news streaming," in *Proceedings of Text2Story - Third Workshop on Narrative Extraction From Texts co-located with 42nd European Conference on Information Retrieval*, 2020.

[138] J. Santos, A. Mendes, and S. Miranda, "Simplifying multilingual news clustering through projection from a shared space," in *Proceedings of Text2Story — Fifth Workshop on Narrative Extraction From Texts held in conjunction with the 44th European Conference on Information Retrieval*, 2022.

[139] L. Zhang, N. Xiao, W. Yang, and J. Li, "Advanced heterogeneous feature fusion machine learning models and algorithms for improving indoor localization," *Sensors (Basel, Switzerland)*, vol. 19, p. 125, 2019. DOI: 10.3390/s19010125.

[140] M.-R. Bouguelia, A. Karlsson, S. Pashami, S. Nowaczyk, and A. Holst, "Mode tracking using multiple data streams," *An international journal on information fusion*, vol. 43, pp. 33–46, 2018. DOI: 10.1016/j.inffus.2017.11.011.

[141] J.-L. Kong, Z.-N. Wang, X.-B. Jin, X.-Y. Wang, T.-L. Su, and J.-L. Wang, "Semi-supervised segmentation framework based on spot-divergence supervoxelization of multi-sensor fusion data for autonomous forest machine applications," *Sensors (Basel, Switzerland)*, vol. 18, 2018. DOI: 10.3390/s18093061.

[142] J. Wu, Y. Feng, and P. Sun, "Sensor fusion for recognition of activities of daily living," *Sensors (Basel, Switzerland)*, vol. 18, p. 4029, 2018. DOI: 10.3390/s18114029.

[143] M. Ma, Q. Song, Y. Gu, Y. Li, and Z. Zhou, "An adaptive zero velocity detection algorithm based on multi-sensor fusion for a pedestrian navigation system," *Sensors (Basel, Switzerland)*, vol. 18, p. 3261, 2018. DOI: 10.3390/s18103261.

[144] Y. Zhou and W. Xue, "A multisensor fusion method for tool condition monitoring in milling," *Sensors (Basel, Switzerland)*, vol. 18, p. 3866, 2018. DOI: 10.3390/s18113866.

[145] P. Shi, G. Li, Y. Yuan, and L. Kuang, "Data fusion using improved support degree function in aquaculture wireless sensor networks," *Sensors (Basel, Switzerland)*, vol. 18, p. 3851, 2018. DOI: 10.3390/s18113851.

[146] F. Zhou, P. Hu, S. Yang, and C. Wen, "A multimodal feature fusion-based deep learning method for online fault diagnosis of rotating machinery," *Sensors (Basel, Switzerland)*, vol. 18, p. 3521, 2018. DOI: 10.3390/s18103521.

[147] K. Lu, L. Yang, F. Seoane, F. Abtahi, M. Forsman, and K. Lindecrantz, "Fusion of heart rate, respiration and motion measurements from a wearable sensor system to enhance energy expenditure estimation," *Sensors (Basel, Switzerland)*, vol. 18, 2018. DOI: 10.3390/s18093092.

[148] J. Hu, T. Huang, J. Zhou, and J. Zeng, "Electronic systems diagnosis fault in gasoline engines based on multi-information fusion," *Sensors (Basel, Switzerland)*, vol. 18, 2018. DOI: 10.3390/s18092917.

[149] B. Wu, T. Huang, Y. Jin, J. Pan, and K. Song, "Fusion of high-dynamic and low-drift sensors using kalman filters," *Sensors (Basel, Switzerland)*, vol. 19, p. 186, 2019. DOI: 10.3390/s19010186.

[150] A. Akbar, G. Kousiouris, H. Pervaiz, *et al.*, "Real-time probabilistic data fusion for large-scale IoT applications," *IEEE access: practical innovations, open solutions*, vol. 6, pp. 10 015–10 027, 2018. DOI: 10.1109/access.2018.2804623.

[151] K. Zhang, X. R. Li, and Y. Zhu, "Optimal update with out-of-sequence measurements," *IEEE transactions on signal processing: a publication of the IEEE Signal Processing Society*, vol. 53, pp. 1992–2004, 2005. DOI: 10.1109/tsp.2005.847830.

[152] B. Khaleghi, A. Khamis, and F. Karray, "Multisensor data fusion," in *Multisensor Data Fusion*, CRC Press, 2017, pp. 15–33. DOI: 10.1201/b18851-2.

[153] D. Lahat, T. Adali, and C. Jutten, "Multimodal data fusion: An overview of methods, challenges, and prospects," *Proceedings of the IEEE. Institute of Electrical and Electronics Engineers*, vol. 103, pp. 1449–1477, 2015. DOI: 10.1109/jproc.2015.2460697.

[154] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, and F. Herrera, "A survey on data preprocessing for data stream mining: Current status and future directions," *Neurocomputing*, vol. 239, pp. 39–57, 2017. DOI: 10.1016/j.neucom.2017.01.078.

[155] S. García, S. Ramírez-Gallego, J. Luengo, J. M. Benítez, and F. Herrera, "Big data preprocessing: Methods and prospects," *Big data analytics*, vol. 1, 2016. DOI: 10.1186/s41044-016-0014-0.

[156] I. Zliobaite and B. Gabrys, "Adaptive preprocessing for streaming data," *IEEE transactions on knowledge and data engineering*, vol. 26, pp. 309–321, 2014. DOI: 10.1109/tkde.2012.147.

[157] C. Wang, M. Zhang, F. Shi, P. Xue, and Y. Li, "A hybrid multimodal data fusion-based method for identifying gambling websites," *Electronics*, vol. 11, p. 2489, 2022. DOI: 10.3390/electronics11162489.

[158] D. Kang, V. Gangal, A. Lu, Z. Chen, and E. Hovy, "Detecting and explaining causes from text for a time series event," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2017. DOI: 10.18653/v1/d17-1292.

[159] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine learning research: JMLR*, vol. 3, pp. 993–1022, 2003. DOI: 10.5555/944919.944937.

[160] N. Kanungsukkasem and T. Leelanupab, "Financial latent dirichlet allocation (FinLDA): Feature extraction in text and data mining for financial time series prediction," *IEEE access: practical innovations, open solutions*, vol. 7, pp. 71 645–71 664, 2019. DOI: 10.1109/access.2019.2919993.

[161] A. Sameer El Khatib, "Machine learning and finance: A review using latent dirichlet allocation technique (LDA)," *SSRN Electronic Journal*, 2020. DOI: 10.2139/ssrn.3730256.

[162] M. Torkar and D. Mladenic, "Characterizing financial markets from the event driven perspective," *Applied network science*, vol. 6, 2021. DOI: 10.1007/s41109-021-00417-z.

[163] I. Gurrib and F. Kamalov, "Predicting bitcoin price movements using sentiment analysis: A machine learning approach," *Studies in economics and finance*, vol. ahead-of-print, pp. 347–364, 2021. DOI: 10.1108/sef-07-2021-0293.

[164] M. Shamisavi and A. Jahanshahi, "Forecasting tehran stock exchange trend with time series analysis, fundamental data, and sentiment analysis in news," in *2022 30th International Conference on Electrical Engineering (ICEE)*, IEEE, 2022. DOI: 10.1109/icee55646.2022.9827232.

[165] S. Fakharchian, "Designing a forecasting assistant of the bitcoin price based on deep learning using market sentiment analysis and multiple feature extraction," *Soft computing*, vol. 27, pp. 18 803–18 827, 2023. DOI: 10.1007/s00500-023-09028-5.

[166] C. Hokamp, D. Ghalandari, and P. Ghaffari, "News signals: An NLP library for text and time series," in *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*, Empirical Methods in Natural Language Processing, 2023. DOI: 10.18653/v1/2023.nlposs-1.21.

[167] M. Stonebraker, U. Çetintemel, and S. Zdonik, "The 8 requirements of real-time stream processing," *SIGMOD record*, vol. 34, pp. 42–47, 2005. DOI: 10.1145/1107499.1107504.

[168] C. C. Aggarwal, *Data streams: Models and algorithms*, 2007th ed., C. C. Aggarwal, Ed., ser. Advances in Database Systems. Springer, 2006. DOI: 10.1007/978-0-387-47534-9.

[169] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining data streams: A review," *SIGMOD record*, vol. 34, pp. 18–26, 2005. DOI: 10.1145/1083784.1083789.

[170] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2000, pp. 71–80. DOI: 10.1145/347090.347107.

[171] C. Manapragada, G. I. Webb, and M. Salehi, "Extremely fast decision tree," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, 2018. DOI: 10.1145/3219819.3220005.

[172] N. Kourtellis, G. De Francisci Morales, A. Bifet, and A. Murdopo, "VHT: Vertical hoeffding tree," in *2016 IEEE International Conference on Big Data (Big Data)*, IEEE, 2016. DOI: 10.1109/bigdata.2016.7840687.

[173] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural networks: the official journal of the International Neural Network Society*, vol. 113, pp. 54–71, 2019. DOI: 10.1016/j.neunet.2019.01.012.

[174] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735.

[175] Q. Wen, T. Zhou, C. Zhang, *et al.*, "Transformers in time series: A survey," in *International Joint Conference on Artificial Intelligence(IJCAI)*, 2023.

[176] H. Zhou, S. Zhang, J. Peng, *et al.*, "Informer: Beyond efficient transformer for long sequence time-series forecasting," *Proceedings of the ... AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence*, vol. 35, pp. 11 106–11 115, 2021. DOI: 10.1609/aaai.v35i12.17325.

[177] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," *Advances in Neural Information Processing Systems*, vol. 34, pp. 22 419–22 430, 2021.

[178] K. C. Chen, L. Dicker, C. Eisenach, and D. Madeka, "MQTransformer: Multi-horizon forecasts with context dependent attention and optimal bregman volatility," in *KDD 2022 Workshop on Mining and Learning from Time Series – Deep Forecasting: Models, Interpretability, and Applications*, 2022.

[179] S.-A. Chen, C.-L. Li, N. Yoder, S. O. Arik, and T. Pfister, "TSMixer: An all-MLP architecture for time series forecasting," *arXiv [cs.LG]*, 2023.

[180] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," in *The Eleventh International Conference on Learning Representations*, 2022.

[181] Y. Liu, T. Hu, H. Zhang, *et al.*, "ITransformer: Inverted transformers are effective for time series forecasting," *arXiv [cs.LG]*, 2023.

[182] H. Emami, X.-H. Dang, Y. Shah, and P. Zerfos, "Modality-aware transformer for time series forecasting," *arXiv [cs.LG]*, 2023.

[183] A. Garza and M. Mergenthaler-Canseco, "TimeGPT-1," *arXiv [cs.LG]*, 2023.

[184] N. Gruver, M. Finzi, S. Qiu, and A. G. Wilson, "Large language models are zero-shot time series forecasters," *arXiv [cs.LG]*, 2023.

[185] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, 2020. DOI: 10.1145/3394486.3403118.

[186] M. Jin, H. Y. Koh, Q. Wen, *et al.*, "A survey on graph neural networks for time series: Forecasting, classification, imputation, and anomaly detection," *arXiv [cs.LG]*, 2023.

[187] C. Villani, *Optimal Transport*, ser. Grundlehren der mathematischen Wissenschaften. Springer, 2016.

[188] L. V. Kantorovich, "On the translocation of masses," *Journal of mathematical sciences*, vol. 133, pp. 1381–1382, 2006. DOI: 10.1007/s10958-006-0049-2.

[189] G. Peyré and M. Cuturi, "Computational optimal transport: With applications to data science," *Foundations and Trends® in Machine Learning*, vol. 11, pp. 355–607, 2019. DOI: 10.1561/2200000073.

[190] R. M. Dudley, "Distances of probability measures and random variables," in *Selected Works of R.M. Dudley*, Springer New York, 2010, pp. 28–37. DOI: 10.1007/978-1-4419-5821-1_4.

[191] A. L. Gibbs and F. E. Su, "On choosing and bounding probability metrics," *Revue internationale de statistique [International statistical review]*, vol. 70, pp. 419–435, 2002. DOI: 10.1111/j.1751-5823.2002.tb00178.x.

[192] E. Novak, L. Bizjak, D. Mladenić, and M. Grobelnik, "Why is a document relevant? understanding the relevance scores in cross-lingual document retrieval," *Knowledge-based systems*, vol. 244, p. 108 545, 2022. DOI: 10.1016/j.knosys.2022.108545.

[193] B. Su and G. Hua, "Order-preserving optimal transport for distances between sequences," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, pp. 2961–2974, 2019. DOI: 10.1109/TPAMI.2018.2870154.

[194] E. Novak, L. Bizjak, D. Mladenić, and M. Grobelnik, "Evaluating text generation model performance by combining semantic meaning and word order," *IEEE Access*, vol. 12, pp. 95 265–95 277, 2024. DOI: 10.1109/ACCESS.2024.3426082.

[195] A. Paszke, S. Gross, F. Massa, *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2019.

[196] E. Novak, *The LM-EMD model code*, https://github.com/eriknovak/model-LM-EMD, (accessed 24 April 2024), 2021.

[197] T. Wolf, L. Debut, V. Sanh, *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Association for Computational Linguistics, 2020, pp. 38–45. DOI: 10.18653/v1/2020.emnlp-demos.6.

[198] Y. Wu, M. Schuster, Z. Chen, *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv [cs.CL]*, 2016.

[199] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv [cs.LG]*, 2017.

[200] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "RCV1: A new benchmark collection for text categorization research," *Journal of machine learning research: JMLR*, vol. 5, pp. 361–397, 2004.

[201] P. Bajaj, D. Campos, N. Craswell, *et al.*, "MS MARCO: A human generated MAchine reading COmprehension dataset," in *Proceedings of the Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches 2016 Co-Located with the 30th Annual Conference on Neural Information Processing Systems*, NIPS, 2016.

[202] F. Nanni, B. Mitra, M. Magnusson, and L. Dietz, "Benchmark for complex answer retrieval," in *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, ACM, 2017. DOI: 10.1145/3121050.3121099.

[203] S. Schamoni, F. Hieber, A. Sokolov, and S. Riezler, "Learning translational and knowledge-based similarities from relevance rankings for cross-language retrieval," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Association for Computational Linguistics, 2014, pp. 488–494. DOI: 10.3115/v1/p14-2080.

[204] S. Sasaki, S. Sun, S. Schamoni, K. Duh, and K. Inui, "Cross-lingual learning-to-rank with shared representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, Association for Computational Linguistics, 2018. DOI: 10.18653/v1/n18-2073.

[205] S. Sun and K. Duh, "CLIRMatrix: A massively large collection of bilingual and multilingual datasets for cross-lingual information retrieval," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, 2020. DOI: 10.18653/v1/2020.emnlp-main.340.

[206] H. Schwenk, V. Chaudhary, S. Sun, H. Gong, and F. Guzmán, "WikiMatrix: Mining 135M parallel sentences in 1620 language pairs from wikipedia," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, Association for Computational Linguistics, 2021. DOI: 10.18653/v1/2021.eacl-main.115.

[207] I. Vulić and M.-F. Moens, "Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, 2015. DOI: 10.1145/2766462.2767752.

[208] Facebook Inc., *Aligned word vectors · fasttext*, https://fasttext.cc/docs/en/aligned-vectors.html, (accessed 24 April 2024), 2020.

[209] R. Flamary and N. Courty, *POT: Python Optimal Transport — POT Python Optimal Transport 0.7.0 documentation*, https://pythonot.github.io/, (accessed 24 April 2024), 2017.

[210] R. Al-Rfou, B. Perozzi, and S. Skiena, "Polyglot: Distributed word representations for multilingual NLP," *arXiv [cs.CL]*, 2013.

[211] J. Liu, X. Zhang, D. Goldwasser, and X. Wang, "Cross-lingual document retrieval with smooth learning," *arXiv [cs.IR]*, 2020.

[212] H. P. Luhn, "A statistical approach to mechanized encoding and searching of literary information," *IBM journal of research and development*, vol. 1, pp. 309–317, 1957. DOI: 10.1147/rd.14.0309.

[213] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," in *Document retrieval systems*, Taylor Graham Publishing, 1988, pp. 132–142. DOI: 10.5555/106765.106782.

[214] E. Novak, *The OPWScore metric code*, https://github.com/eriknovak/metric-OPWScore, (accessed 24 April 2024), 2023.

[215] A. Williams, N. Nangia, and S. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 1112–1122. DOI: 10.18653/v1/N18-1101.

[216] B. Ondřej, Y. Graham, and A. Kamran, "Results of the WMT17 metrics shared task," in *Proceedings of the Second Conference on Machine Translation*, Association for Computational Linguistics, 2017, pp. 489–513.

[217] O. Bojar, Y. Graham, A. Kamran, and M. Stanojević, "Results of the WMT16 metrics shared task," in *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, Association for Computational Linguistics, 2016, pp. 199–231. DOI: 10.18653/v1/w16-2302.

[218] Q. Ma, O. Bojar, and Y. Graham, "Results of the WMT18 metrics shared task," in *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, Association for Computational Linguistics, 2018. DOI: 10.18653/v1/W18-64077.

[219] N. Mathur, J. Wei, M. Freitag, Q. Ma, and B. Ondřej, "Results of the WMT20 metrics shared task," in *Proceedings of the Fifth Conference on Machine Translation*, Association for Computational Linguistics, 2020, pp. 688–725.

[220] M. Freitag, R. Rei, N. Mathur, *et al.*, "Results of the WMT21 metrics shared task: Evaluating metrics with expert-based human evaluations on TED and news domain," in *Proceedings of the Sixth Conference on Machine Translation*, Association for Computational Linguistics, 2021, pp. 733–774.

[221] E. Novak, E. Calcina, D. Mladenić, and M. Grobelnik, *The news articles reporting on the 2021 tokyo olympics data set OG2021 (public)*, Slovenian language resource repository CLARIN.SI, 2024. [Online]. Available: http://hdl.handle.net/11356/1922.

[222] E. Novak, E. Calcina, D. Mladenić, and M. Grobelnik, *The news articles reporting on the 2021 tokyo olympics data set OG2021 (research)*, Slovenian language resource repository CLARIN.SI, 2024. [Online]. Available: http://hdl.handle.net/11356/1921.

[223] E. Novak, E. Calcina, D. Mladenić, and M. Grobelnik, "The 2021 tokyo olympics multilingual news article dataset," *In Review*, 2024.

[224] E. Novak, D. Mladenić, and M. Grobelnik, "Online multilingual news clustering using wasserstein distance," *In Review*, 2024.

[225] D. Greene and P. Cunningham, "Practical solutions to the problem of diagonal dominance in kernel document clustering," in *Proc. 23rd International Conference on Machine learning (ICML'06)*, ACM Press, 2006, pp. 377–384.

[226] L. Gebhard and F. Hamborg, "The POLUSA dataset: 0.9M political news articles balanced by time and outlet popularity," in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*, ACM, 2020. DOI: 10.1145/3383583.3398567.

[227] D. Lewis, *Reuters-21578 text categorization collection*, 1997. DOI: 10.24432/C52G6M.

[228] R. Misra and J. Grover, *Sculpting Data for ML: The first act of Machine Learning*. Independently Published, 2021.

[229] R. Misra, "News category dataset," *arXiv [cs.CL]*, 2022.

[230] J. Nørregaard, B. D. Horne, and S. Adalı, "NELA-GT-2018: A large multi-labelled news dataset for the study of misinformation in news articles," *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 13, pp. 630–638, 2019. DOI: 10.1609/icwsm.v13i01.3261.

[231] M. Gruppi, B. D. Horne, and S. Adalı, "NELA-GT-2019: A large multi-labelled news dataset for the study of misinformation in news articles," *arXiv [cs.CY]*, 2020.

[232] J. Tiedemann and N. Ljubešić, "Efficient discrimination between closely related languages," in *Proceedings of COLING 2012*, 2012, pp. 2619–2634.

[233] H. Baradaran Hashemi, A. Shakery, and H. Faili, "Creating a persian-english comparable corpus," in *Multilingual and Multimodal Information Access Evaluation*, ser. Lecture notes in computer science, Springer Berlin Heidelberg, 2010, pp. 27–39. DOI: 10.1007/978-3-642-15998-5_5.

[234] R. Hauser, J. Vamvas, S. Ebling, and M. Volk, "A multilingual simplified language news corpus," in *Proceedings of the 2nd Workshop on Tools and Resources to Empower People with REAding DIfficulties (READI) within the 13th Language Resources and Evaluation Conference*, 2022, pp. 25–30.

[235] D. Varab and N. Schluter, "MassiveSumm: A very large-scale, very multilingual, news summarisation dataset," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2021, pp. 10 150–10 161. DOI: 10.18653/v1/2021.emnlp-main.797.

[236] J. Mackenzie, R. Benham, M. Petri, J. R. Trippas, J. S. Culpepper, and A. Moffat, "CC-news-en: A large english news corpus," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, ACM, 2020. DOI: 10.1145/3340531.3412762.

[237] F. Leeb and B. Schölkopf, "A diverse multilingual news headlines dataset from around the world," *arXiv [cs.CL]*, 2024.

[238] E. Novak and E. Calcina, *The OG2021 data set creation code*, https://github.com/E3-JSI/dataset-OG2021, (accessed 29 April 2024), 2024.

[239] E. Novak, "News stream clustering using multilingual language models," in *Proceedings of Slovenian KDD Conference on Data Mining and Data Warehouses (SiKDD)*, 2021.

[240] N. Thakur, N. Reimers, J. Daxenberger, and I. Gurevych, "Augmented SBERT: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, 2020. DOI: 10.18653/v1/2021.naacl-main.28.

[241] S. Tedeschi, V. Maiorca, N. Campolungo, F. Cecconi, and R. Navigli, "WikiNEuRal: Combined neural and knowledge-based silver data creation for multilingual NER," in *Findings of the Association for Computational Linguistics: EMNLP 2021*, Association for Computational Linguistics, 2021, pp. 2521–2533. DOI: 10.18653/v1/2021.findings-emnlp.215.

[242] T. Aarsen, "SpanMarker for named entity recognition," M.S. thesis, Radboud University, 2023.

[243] S. Tedeschi and R. Navigli, "MultiNERD: A multilingual, multi-genre and fine-grained dataset for named entity recognition (and disambiguation)," in *Findings of the Association for Computational Linguistics: NAACL 2022*, Association for Computational Linguistics, 2022, pp. 801–812. DOI: 10.18653/v1/2022.findings-naacl.60.

[244] E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo, "A comparison of extrinsic clustering evaluation metrics based on formal constraints," *Information retrieval*, vol. 12, pp. 461–486, 2009. DOI: 10.1007/s10791-008-9066-8.

[245] T. Joachims, "Optimizing search engines using clickthrough data," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2002. DOI: 10.1145/775047.775067.

[246] E. Novak, "News stream clustering using multilingual language models," in *Proceedings of Slovenian KDD Conference on Data Mining and Data Warehouses (SiKDD)*, Ljubljana, Slovenia, 2021.

[247] K. Kenda, B. Kažič, E. Novak, and D. Mladenić, "Streaming data fusion for the internet of things," *Sensors*, vol. 19, p. 1955, 8 2019. DOI: 10.3390/s19081955.

[248] L. Rodríguez-Mazahua, C.-A. Rodríguez-Enríquez, J. L. Sánchez-Cervantes, J. Cervantes, J. L. García-Alcaraz, and G. Alor-Hernández, "A general perspective of big data: Applications, tools, challenges and trends," *The journal of supercomputing*, vol. 72, pp. 3073–3113, 2016. DOI: 10.1007/s11227-015-1501-1.

[249] E. Ahmed, I. Yaqoob, I. A. T. Hashem, *et al.*, "The role of big data analytics in internet of things," *Computer networks*, vol. 129, pp. 459–471, 2017. DOI: 10.1016/j.comnet.2017.06.013.

[250] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys*, vol. 46, pp. 1–37, 2014. DOI: 10.1145/2523813.

[251] A. Gepperth and B. Hammer, "Incremental learning algorithms and applications," in *European Symposium on Artificial Neural Networks (ESANN)*, 2016.

[252] A. Bifet, G. Holmes, B. Pfahringer, *et al.*, "MOA: Massive online analysis, a framework for stream classification and clustering," in *Proceedings of the First Workshop on Applications of Pattern Analysis*, PMLR, 2010, pp. 44–50.

[253] J. Manyika, M. Chui, P. Bisson, *et al.*, "Unlocking the potential of the internet of things," Tech. Rep., 2015.

[254] D. Q. Tu, A. S. M. Kayes, W. Rahayu, and K. Nguyen, "ISDI: A new window-based framework for integrating IoT streaming data from multiple sources," in *Advanced Information Networking and Applications*, ser. Advances in intelligent systems and computing, Springer International Publishing, 2020, pp. 498–511. DOI: 10.1007/978-3-030-15032-7_42.

[255] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE transactions on knowledge and data engineering*, vol. 26, pp. 97–107, 2014. DOI: 10.1109/tkde.2013.109.

[256] C. Tekin, L. Canzian, and M. van der Schaar, "Context-adaptive big data stream mining," in *2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, 2014, pp. 483–490. DOI: 10.1109/allerton.2014.7028494.

[257] K. Kenda, M. Skrjanc, and A. Borstnik, "Modelling of the complex data space: Architecture and use cases from NRG4CAST project," in *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*, IEEE, 2015. DOI: 10.1109/iisa.2015.7388056.

[258] M. Christ, A. W. Kempa-Liehr, and M. Feindt, "Distributed and parallel time series feature extraction for industrial big data applications," *arXiv [cs.LG]*, 2016.

[259] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time series FeatuRe extraction on basis of scalable hypothesis tests (tsfresh – a python package)," *Neurocomputing*, vol. 307, pp. 72–77, 2018. DOI: 10.1016/j.neucom.2018.03.067.

[260] E. Novak, *The LSTM text extension code*, https://github.com/eriknovak/model-lstm-text-extension, (accessed 29 April 2024), 2024.

[261] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv [cs.LG]*, 2014.

[262] L. N. Smith, "Cyclical learning rates for training neural networks," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, 2017, pp. 464–472. DOI: 10.1109/wacv.2017.58.

[263] H. Xu, L. Chai, Z. Luo, and S. Li, "Stock movement prediction via gated recurrent unit network based on reinforcement learning with incorporated attention mechanisms," *Neurocomputing*, vol. 467, pp. 214–228, 2022. DOI: 10.1016/j.neucom.2021.09.072.

[264] E. Novak, *The WAC algorithm code*, https://github.com/eriknovak/WAC, (accessed 29 April 2024), 2024.

[265] R. Al-Ghezi and M. Kurimo, "Graph-based syntactic word embeddings," in *Proceedings of the Graph-based Methods for Natural Language Processing (TextGraphs)*, Association for Computational Linguistics, 2020, pp. 72–78. DOI: 10.18653/v1/2020.textgraphs-1.8.

Bibliography

Publications Related to the Thesis

Journal Articles

E. Novak, L. Bizjak, D. Mladenić, and M. Grobelnik, "Why is a document relevant? understanding the relevance scores in cross-lingual document retrieval," *Knowledge-based systems*, vol. 244, p. 108 545, 2022. DOI: 10.1016/j.knosys.2022.108545.

E. Novak, L. Bizjak, D. Mladenić, and M. Grobelnik, "Evaluating text generation model performance by combining semantic meaning and word order," *IEEE Access*, vol. 12, pp. 95 265–95 277, 2024. DOI: 10.1109/ACCESS.2024.3426082.

K. Kenda, B. Kažič, E. Novak, and D. Mladenić, "Streaming data fusion for the internet of things," *Sensors*, vol. 19, p. 1955, 8 2019. DOI: 10.3390/s19081955.

E. Novak, E. Calcina, D. Mladenić, and M. Grobelnik, "The 2021 tokyo olympics multilingual news article dataset," *In Review*, 2024.

E. Novak, D. Mladenić, and M. Grobelnik, "Online multilingual news clustering using wasserstein distance," *In Review*, 2024.

Conference Papers

A. Mladenić Grobelnik, E. Novak, D. Mladenić, and M. Grobelnik, "Slomet – slovenian commonsense description," in *Proceedings of Slovenian KDD Conference on Data Mining and Data Warehouses (SiKDD)*, Ljubljana, Slovenia, 2022.

E. Novak, "News stream clustering using multilingual language models," in *Proceedings of Slovenian KDD Conference on Data Mining and Data Warehouses (SiKDD)*, Ljubljana, Slovenia, 2021.

E. Novak, J. Urbančič, and M. Jenko, "Preparing multi-modal data for natural language processing," in *Proceedings of Slovenian KDD Conference on Data Mining and Data Warehouses (SiKDD)*, Ljubljana, Slovenia, 2018.

Published Data Sets

E. Novak, E. Calcina, D. Mladenić, and M. Grobelnik, *The news articles reporting on the 2021 tokyo olympics data set OG2021 (public)*, Slovenian language resource repository CLARIN.SI, 2024. [Online]. Available: http://hdl.handle.net/11356/1922.

E. Novak, E. Calcina, D. Mladenić, and M. Grobelnik, *The news articles reporting on the 2021 tokyo olympics data set OG2021 (research)*, Slovenian language resource repository CLARIN.SI, 2024. [Online]. Available: http://hdl.handle.net/11356/1921.

A. Mladenić Grobelnik, E. Novak, D. Mladenić, and M. Grobelnik, *Slovene translation of the atomic 2020 data set SloATOMIC 2020*, Slovenian language resource repository CLARIN.SI, 2022. [Online]. Available: http://hdl.handle.net/11356/1724.

Other Publications

Conference Papers

E. Calcina and E. Novak, "Measuring the similarity of song artists using topic modelling," in *Proceedings of Slovenian KDD Conference on Data Mining and Data Warehouses (SiKDD)*, Ljubljana, Slovenia, 2022.

S. K. Tratnik and E. Novak, "Automatically generating text from film material – a comparison of three models," in *Proceedings of Slovenian KDD Conference on Data Mining and Data Warehouses (SiKDD)*, Ljubljana, Slovenia, 2022.

A. Sunar, E. Novak, and D. Mladenić, "Users' learning pathways on cross-site open educational resources," in *12th International Conference on Computer Supported Education, CSEDU 2020*, SciTePress, 2020.

M. Massri, S. Brezec, E. Novak, and K. Kenda, "Semantic enrichment and analysis of legal domain documents," in *Proceedings of Slovenian KDD Conference on Data Mining and Data Warehouses (SiKDD)*, Ljubljana, Slovenia, 2019.

S. Kralj, Ž. Urbančič, E. Novak, and K. Kenda, "Document embedding models on environmental legal documents," in *Proceedings of Slovenian KDD Conference on Data Mining and Data Warehouses (SiKDD)*, Ljubljana, Slovenia, 2019.

E. Novak and I. Novalija, "Connecting professional skill demand with supply," in *Proceedings of Slovenian KDD Conference on Data Mining and Data Warehouses (SiKDD)*, Ljubljana, Slovenia, 2017.

Biography

The author of this thesis was born on the 24$^{\text{th}}$ of September 1991 in Ljubljana, Slovenia. He attended primary school in his home town in Vrhnika, and secondary school in Ljubljana. In 2010, he enrolled in the single-cycle master's program in Mathematics Education at the Faculty of Mathematics and Physics of the University of Ljubljana, Slovenia. In 2016, he successfully obtained his Master's degree by defending his Master's thesis entitled "Non-negative matrix factorization" under the supervision of Prof. Dr. Bor Plestenjak and co-supervision of Assist. Dr. Andrej Muhič.

In 2016, he enrolled in the Information and Communication Technologies PhD program at the Jožef Stefan International Postgraduate School in Ljubljana, Slovenia, under the supervision of Prof. Dr. Dunja Mladenić. His research work was also carried out in the scope of the European Union Commission-funded projects at the Department for Artificial Intelligence at the Jožef Stefan Institute, Ljubljana, Slovenia. During this time, he also gave a number of presentations on practical and ethical use of artificial intelligence models at various conferences and events, and took on the role of a mentor, providing guidance and support to younger students interested in artificial intelligence, machine learning and natural language processing.

His research interests are in artificial intelligence and machine learning, with a focus on natural language processing and data stream analysis. He works on techniques for analyzing sequential textual data, with a focus on similarity measures, information extraction, and explainability and transparency of the developed methods.