

MACHINE LEARNING OF SEMANTICS FOR TEXT UNDERSTANDING

Janez Starc

Doctoral Dissertation
Jožef Stefan International Postgraduate School
Ljubljana, Slovenia

Supervisor: Prof. Dr. Dunja Mladenić, Jožef Stefan Institute and Jožef Stefan International Postgraduate School, Ljubljana, Slovenia

Evaluation Board:

Prof. Dr. Tomaž Erjavec, Chair, Jožef Stefan Institute, Ljubljana, Slovenia

Asst. Prof. Dr. Primož Škraba, Member, Jožef Stefan Institute, Ljubljana, Slovenia

Prof. Dr. Estevam Rafael Hruschka, Member, University of São Carlos, São Carlos, São Paulo, Brazil

MEDNARODNA PODIPLOMSKA ŠOLA JOŽEFA STEFANA
JOŽEF STEFAN INTERNATIONAL POSTGRADUATE SCHOOL



Janez Starc

MACHINE LEARNING OF SEMANTICS FOR TEXT UN-
DERSTANDING

Doctoral Dissertation

STROJNO UČENJE SEMANTIKE ZA RAZUMEVANJE
BESEDILA

Doktorska disertacija

Supervisor: Prof. Dr. Dunja Mladenić

Ljubljana, Slovenia, February 2017

Acknowledgments

First, I would like to thank my supervisor Professor Dunja Mladenić for her support and guidance throughout my doctoral studies. I would also like to thank the members of my doctoral evaluation board, Primož Škraba, Tomaž Erjavec, and Estevam Rafael Hruschka for their insightful comments. I am grateful to my colleagues from the Jožef Stefan Institute for their contributions, valuable discussions and for making my time during the studies very enjoyable. Special thanks goes to my partner for her support and encouragement. Last but not least, I would like to thank my parents for their unconditional support throughout my life.

This work was supported by the Slovenian Research Agency and the ICT Programme of the EC under XLike (ICT-STREP-288342) and XLime (FP7-ICT-611346).

Abstract

Text Understanding is a long term goal of Artificial Intelligence. Its mission is to create algorithms that will fully automatically understand human-composed text. In this dissertation we propose, implement and evaluate machine learning approaches and meaning representations to make progress towards Text Understanding. Meaning representations are used to encode the information contained in a sentence. Furthermore, they can be used in applications that require detailed understanding of the sentence. The focus of our work is on the intrinsic meaning (semantics) of sentences rather than their context or syntactic characteristics.

An ontology is a conceptualization that can be used to express the meaning of text. Predicate logic, which connects concepts from the ontology, is one of the meaning representations that we use. We propose two rule-based approaches for mapping text into predicate logic. One approach is based on textual patterns, while the other approach is based on a syntax parser. Rules are manually constructed and the approaches work on the assumption of a complete ontology. This led us to develop a grammar induction approach for semantic parsing and ontology learning. The induced context-free grammar parses a sentence of text into a semantic tree, which is a meaning representation, where each node has its own semantic category, e.g. person, location, profession. Furthermore, some of the nodes can be aligned with ontology concepts. The trees are used as a source for relation extraction and instance discovery, while a taxonomy is extracted from the grammar. Thus, both grammar and semantic trees are used to extend the ontology, so that it also represents the meaning of the input sentences.

Text understanding can be both learned and evaluated through the task of Natural Language Inference (NLI), where the goal is to determine whether one sentence (hypothesis) entails, contradicts or is neutral in respect to the other sentence (premise). Instead of focusing on this classification task, we propose several text generative neural networks for generating hypotheses which allow the construction of new NLI datasets. To evaluate the models, we propose a new metric – the accuracy of the classifier trained on the generated dataset and tested on the original, manually constructed dataset. The accuracy obtained by our best generative model is only 2.7% lower than the accuracy of the classifier trained on the original dataset. The model learns a mapping embedding for each training example. Furthermore, combining the best generated and original dataset results in the highest accuracy. Our metric awards diverse, accurate, non-trivial and comprehensible examples. By comparing various metrics we show that datasets that obtain higher ROUGE or METEOR scores do not necessarily yield higher classification accuracies. We also provide an analysis of what are the characteristics of a good dataset including the indistinguishability of the generated dataset from the original one.

Povzetek

Razumevanje besedila (Text Understanding) je eden dolgorčnih ciljev umetne inteligence. Glavna naloga tega področja je razvoj algoritmov, ki bodo popolnoma samostojno razumeli besedila, ki jih je ustvaril človek. V tej disertaciji so predlagani, implementirani in ovrednoteni različni pristopi na podlagi strojnega učenja ter različne predstavitve pomena. Osredotočamo se predvsem na semantični vidik stavkov, manj pa na kontekst ali na sintaktične značilnosti.

Ontologije so sestavljene iz množice konceptov. Predikatna logika, ki povezuje te koncepte, je ena od predstavitev pomena, ki jih uporabljamo. Predlagamo dva pristopa na podlagi pravil, ki omogočajo preslikavo besedila v predikatno logiko. En pristop je osnovan na različnih vzorcih besedila, drugi pa uporablja sintaktični razčlenjevalnik. Vsa ta pravila so ročno ustvarjena. Postopki pa predpostavljajo popolno ontologijo. Te omejitve so bile povod za razvoj novega pristopa za indukcijo kontekstno neodvisne slovnice, ki omogoča semantično razčlenjevanje in učenje ontologije. Kontekstno neodvisna slovnica stavkov razčleni v semantično drevo. To pa je predstavitev znanja, pri kateri ima vsako vozlišče svojo semantično kategorijo, npr. oseba, barva in poklic. Nekatera vozlišča pa se tudi ujemajo s koncepti iz ontologije. Semantična drevesa so uporabljena za ekstrakcijo relacij in razpoznavanje novih pojmov. Iz slovnice pa je izluščena taksonomija. Torej so tako slovnica kot tudi semantična drevesa uporabljena za nadgradnjo ontologije, tako da le-ta vsebuje tudi pomen vhodnih stavkov.

Razumevanje besedila se lahko nauči in ovrednoti s pomočjo sklepanja v besedilu (Natural Language Inference). Glavni cilj te naloge je ugotoviti, ali nek stavek logično sledi drugemu stavku ali pa je v relaciji z njim kontradiktoren oziroma nevtralen. Namesto da bi se osredotočili na ta klasifikacijski problem, smo se osredotočili na problem generiranja novih hipotez. Predlagamo več generativnih modelov nevronske mreže, ki generirajo nove hipoteze, kar omogoča gradnjo novih naborov podatkov. Za vrednotenje teh modelov predlagamo novo metriko. To je natančnost klasifikatorja, naučenega na novem generiranem naboru podatkov in testiranega na originalnem naboru podatkov, ki jih je ustvaril človek. Natančnost klasifikatorja, ki je bil naučen preko najboljšega generativnega modela, je samo 2.7% nižja od natančnosti klasifikatorja, naučenega na originalnih podatkih. Ta model se nauči vložitev preslikave za vsak učni primer posebej. Če pa združimo najboljše generirane in originalni nabor podatkov za učenje klasifikatorja, pa le-ta doseže najvišjo natančnost. Predlagana metrika nagrajuje raznolike, netrivialne in razumljive učne primere. Ugotovili smo, da nabori podatkov, ki dosežejo visoko oceno z drugimi metrikami, kot sta ROUGE in METEOR, ne dosežejo nujno visoke natančnosti naučenih klasifikatorjev. Analizirali smo tudi značilnosti dobrih naborov podatkov. Ena takih značilnosti je nezmožnost ločevanja generiranega nabora podatkov od originalnega nabora.

Contents

List of Figures	xv
List of Tables	xvii
List of Algorithms	xix
Abbreviations	xxi
1 Introduction	1
1.1 Aims and Hypothesis	3
1.2 Scientific Contributions	4
1.3 Thesis Structure	5
2 Related Work	7
2.1 Meaning Representations	8
2.1.1 Existing ontologies	9
2.2 Learning Approaches	9
2.2.1 Semantic parsing	9
2.2.2 Ontology learning	10
2.2.3 Grammar induction	11
2.2.4 Natural Language Inference	11
2.2.5 Natural Language Generation	12
2.2.6 Neural networks for NLP	12
3 Representations	15
3.1 Text Representations	15
3.1.1 Layered representation	15
3.1.2 Word vector representation	15
3.2 Textual Patterns	16
3.2.1 Grammar approach	16
3.3 Meaning Representations	17
3.3.1 Predicate logic	17
3.3.1.1 Existing ontologies	19
3.3.2 Semantic trees	19
3.3.3 Distributed representation	20
3.3.3.1 Neural networks	21
4 Learning Approaches	23
4.1 Mapping to Predicate Logic with Textual Patterns	23
4.1.1 Creating rules	24
4.1.1.1 Layered generalization	24
4.1.1.2 Finding valuable textual patterns	24

4.2	Mapping to Predicate Logic Using the Syntax Parser	25
4.2.1	Ontology resources	26
4.2.2	Rewrite rules	27
4.3	Grammar Induction	29
4.3.1	Parser	30
4.3.1.1	Ambiguity	30
4.3.1.2	Optimizations	31
4.3.2	Rule induction	32
4.3.2.1	Bottom-up parsing	32
4.3.3	Property assignment	32
4.3.4	Seed rules	33
4.4	Taxonomy Construction from Grammar	33
4.5	Mapping Semantic Trees to Predicate Logic	34
4.5.1	Mapping with rewrite rules	34
4.5.2	Relation extraction from semantic trees	34
4.6	Generating Natural Language Inference Hypotheses	35
4.6.1	Models	37
4.6.1.1	Classification model	37
4.6.1.2	Generative models	38
4.6.1.2.1	Generating hypotheses	41
4.6.1.3	Discriminative model	41
5	Evaluation	43
5.1	Mapping to Predicate Logic with Textual Patterns	43
5.2	Mapping to Predicate Logic Using Syntax Parser	45
5.3	Grammar Induction	46
5.3.1	Datasets	46
5.3.2	Grammar induction experiments	47
5.3.2.1	Results	48
5.3.3	Instance extraction	49
5.3.4	Relation extraction	50
5.4	Dataset Generation	52
5.4.1	Experiment details	52
5.4.2	Preliminary evaluation	53
5.4.3	Other models	58
5.4.4	Qualitative evaluation	58
5.5	Approach Comparison	60
5.5.1	Text preprocessing	60
5.5.2	Concept disambiguation	61
5.5.3	Sentence structure modeling	61
5.5.4	Cardinality	62
5.5.5	Manual work	62
5.5.6	Meaning representations	62
6	Prototype	65
7	Conclusions	69
7.1	Scientific Contributions	70
7.2	Future Work	70
	Appendix A More Generated NLI Examples	73

Contents	xiii
References	77
Bibliography	85
Biography	87

List of Figures

Figure 1.1:	Approaches overview	2
Figure 3.1:	An example of a semantic tree	20
Figure 4.1:	F-structure of the sentence “John drives a car.”	26
Figure 4.2:	Grammar induction process.	30
Figure 4.3:	Extraction of variable trees	36
Figure 4.4:	NLI dataset generation approach.	37
Figure 4.5:	NLI classification model.	38
Figure 4.6:	Generative model architecture	39
Figure 4.7:	ATTEMBEDDECODER model.	41
Figure 5.1:	Statistics of the iterative process.	49
Figure 5.2:	The automaton model for relation extraction.	51
Figure 5.3:	Accuracies of the unfiltered generated datasets	54
Figure 5.4:	Accuracies of classifiers trained on the generated dataset	55
Figure 5.5:	Comparison of unfiltered generated datasets using various metrics	57
Figure 6.1:	The screenshot of the main screen of the prototype.	66
Figure 6.2:	The screenshot of the interface for constructing lexical patterns.	66
Figure 6.3:	The screenshot of the interface for creating rules.	67
Figure 6.4:	The screenshot of the interface for constructing and adding relations.	68

List of Tables

Table 3.1:	Layered representation of a sentence	16
Table 4.1:	Two-step generalization	25
Table 4.2:	Three NLI examples from SNLI corpus.	36
Table 5.1:	Part of the table representing 6-gram patterns and their statistics. . . .	44
Table 5.2:	Selected four mapping rules.	44
Table 5.3:	Evaluation of four selected rules.	45
Table 5.4:	Evaluation of f-structure quality.	45
Table 5.5:	Evaluation of phrase denotation quality.	46
Table 5.6:	Evaluation of meaning representation structure and semantic value of predicates.	46
Table 5.7:	A sample of induced rules.	48
Table 5.8:	Statistics of test set.	48
Table 5.9:	Instance relations per category.	50
Table 5.10:	Performance of various relation extraction models.	52
Table 5.11:	Generated examples to illustrate the proposed approach.	54
Table 5.12:	Comparison of generative models	58
Table 5.13:	The performance of classifiers trained on the original and generated datasets.	59
Table 5.14:	Discrimination error rate on the development set and a sample of 200 examples, evaluated by the discriminative model and human evaluator. .	60

List of Algorithms

Algorithm 4.1: Pseudocode of the main function *parse* of the top-down parser. . . . 31

Abbreviations

CRF	...	Conditional Random Field
LSTM	...	Long Short Term Memory
NLI	...	Natural Language Inference
NLP	...	Natural Language Processing
RDF	...	Resource Description Framework
RNN	...	Recurrent Neural Network

Chapter 1

Introduction

With the digital revolution, an enormous amount of text in digital form is produced every day. This has created a demand for reading and understanding amounts of text that are too large for humans to do it in reasonable time. On the other hand, the advancement of machine learning has created opportunities to develop software that understands text without human supervision. Such software will also lead to many other opportunities in artificial intelligence to develop very intelligent machines.

The purpose of this dissertation is to *develop, implement and evaluate machine learning approaches to text understanding*. The focus of the approaches is on the semantic aspect of language. Semantics is the study of the meaning, compared to syntax, which is concerned with the structure of language and its grammatical correctness, and pragmatics, which also takes into account the context of its utterance.

The proposed approaches process the input text to represent its meaning with a representation that is suitable for further usage in a particular application that requires the understanding of the meaning. This representation is referred to as *meaning representation*. Thus, one of the goals is to learn how to map text to its meaning representation.

Figure 1.1 presents the overview of the dissertation in a graphical form. In this dissertation, predicate logic, semantic trees, and distributed representations are the meaning representations used. Predicate logic is applied here as a formal system that connects concepts that appear in the text to form relations that express the meaning of the text. Concepts and relations are also interconnected to form an ontology, which represents the backbone knowledge. The proposed semantic trees are another representation that connects the concepts, however the concepts are connected in a tree form that models the structure of the sentence. This representation provides In contrast to syntax trees, the nodes of semantic trees are categorized with semantic classes like person, location and profession, instead of syntactical categories like verb, noun, adjective. The proposed representation is a intermediate representation between ontological representation and syntax trees, and allows exploitation of advantages of both approaches. In the distributed representation, the meaning is represented with real vectors, which are in our work, part of neural networks.

We propose several approaches for mapping the text into its meaning representation. One approach is to search for frequent textual patterns and connect them into pattern rules that map text into predicate logic representation. In another rule-based approach the syntax parse tree is transformed into predicate logic. This approach is based on a smaller number of rewrite rules that are coupled with ontological resources.

The evaluation has revealed weaknesses in both of the approaches, especially the large amount of required manual work and poor coverage of rules. To alleviate these weaknesses, we developed a new approach based on grammar induction. This is an extension of the

The proposed generative networks are trained to generate a hypothesis given a premise and a label, which allow us to construct new, unseen examples. Some generative models are built to generate a single optimal response given the input. Such models have been applied to machine translation (Sutskever, Vinyals, & Le, 2014), image caption generation (Xu et al., 2015), or dialogue systems (Serban, Sordoni, Bengio, Courville, & Pineau, 2016). Another type of generative models are autoencoders that generate a stream of random samples from the original distribution. For instance, autoencoders have been used to generate text (Bowman, Vilnis, et al., 2015; J. Li, Luong, & Jurafsky, 2015), and images (Goodfellow et al., 2014). In our setting we combine both approaches to generate a stream of random responses (hypotheses) that comply with the input (premise, label).

But what is a good stream of hypotheses? We argue that a good stream contains *diverse*, *comprehensible*, *accurate* and *non-trivial* hypotheses. A hypothesis is comprehensible if it is grammatical and semantically makes sense. It is accurate if it clearly expresses the relationship (signified by the label) with the premise. Finally, it is non-trivial if it is not trivial to determine the relationship (label) between the hypothesis and premise. For instance, given a premise “A man drives a red car” and label *entailment*, the hypothesis “A man drives a car” is more trivial than “A person is sitting in a red vehicle”.

The next question is how to automatically measure the quality of generated hypotheses. One way is to use standard metrics in text generation tasks, for instance ROUGE (Lin, 2004), BLEU (Papineni, Roukos, Ward, & Zhu, 2002), METEOR (Denkowski & Lavie, 2014). These metrics estimate the similarity between the generated text and the original reference text. In our task they can be used by comparing the generated and reference hypotheses with the same premise and label. The main issue with these metrics is that they penalize diversity, since they penalize the generated hypotheses that are dissimilar to the reference hypothesis. An alternative metric is to use a NLI classifier to test the generated hypothesis if the input label is correct with respect to the premise. A perfect classifier would not penalize diverse hypotheses and would reward accurate and (arguably to some degree) comprehensible hypotheses. However, it would not reward non-trivial hypotheses. To account for the trivialness of the hypothesis, we propose the following evaluation procedure: construct a new NLI dataset from the generated hypotheses, train a NLI classifier on the new dataset, then measure the accuracy of the classifier. If the constructed dataset contains lots of non-trivial examples, it enables the model to learn to classify better, which results in the higher accuracy of the classifier on the testing dataset.

1.1 Aims and Hypothesis

The general aim of this dissertation is to *propose, apply and evaluate several approaches for text understating using machine learning techniques*. The aims are further broken down into the following tasks:

- Propose and evaluate the methodology for mapping text into predicate logic using textual patterns and syntax parser.
- Propose and evaluate the methodology that extends the textual patterns into a grammar induction approach.
- Propose and evaluate the methodology for ontology construction from grammar and from semantic trees.
- Propose and evaluate the methodology for generating NLI hypotheses using generative neural networks.

- Compare the proposed meaning representations and mapping approaches.

In this dissertation we address the following hypotheses that are developed and experimentally tested:

- The proposed semantic trees provide a suitable representation for expressing the meaning of the text as needed for relation and instance extraction.
- Context-free grammar induced from text can be utilized by the proposed novel method for constructing the taxonomic part of the ontology
- Generative neural networks can be used to generate NLI datasets
- A good dataset for training a NLI classifier consists of a variety of accurate, non-trivial and comprehensible examples.

1.2 Scientific Contributions

The contributions of this dissertation are mainly the proposed approaches for Text Understanding and their evaluation. We highlight the following contributions:

- C1** *Context-free grammar induction approach.* The approach allows to build grammars, which are used to parse text into semantic trees and can be utilized by the proposed novel method for constructing the taxonomic part of the ontology.
- C2** *Semantic trees coupled with the proposed ontology learning approaches.* The first approach is based on relation extraction, while the second is based on instance discovery.
- C3** *Text generative neural network.* The network consists of the decoder that learns a mapping embedding for each training example separately.
- C4** *Procedure for the evaluation of generated NLI text hypotheses.* This includes a procedure for constructing a NLI dataset and an evaluation metric for the generated texts – the accuracy of the classifier trained on the generated dataset.

We claim these are contributions to the fields of Natural Language Processing and Machine Learning.

First, we propose a grammar induction approach, which allows joint semantic parsing and ontology learning. The approach is an extension to the approaches that map sentences to predicate logic via textual patterns and syntax parser. The approach not only allows construction of the meaning representation of a sentence (semantic parsing), but also connects the concepts that appear in text with the existing ontology (ontology learning). The novelty of the grammar induction approach is a bootstrapping procedure that uses top-down parsing and bottom-up parsing to find the next most frequent grammar rule. One purpose of the grammar is to extract the taxonomic part of the ontology – classes, instances and their relations. The other purpose is for parsing the sentences into semantic trees, our next contribution. Semantic trees reflect the meaning of sentences. Every node of the tree contains a class and some leaf nodes represent ontology instances. Other leaf nodes are used to find new ontology instances. Semantic trees also serve as the input to the proposed relation extraction method. The method searches the paths between concepts to train a model for predicting new relations.

Our next contribution is neural network models for generating NLI hypotheses. The networks are accompanied by the proposed evaluation metric, which awards diverse, accurate, non-trivial and comprehensible hypotheses, as opposed to other text generation metrics like BLUE, ROUGE and METEOR. We propose several neural networks based on encoder-decoder architecture. The best and most novel architecture learns a mapping embedding for each training example separately instead of an encoder. The accuracy obtained by our best generative model is only 2.7% lower than the accuracy of the classifier trained on the original, manually constructed dataset. While the best accuracy is achieved by the dataset that is combined of the best generated and original dataset.

The code for constructing NLI datasets is published¹, which allows one to easily reproduce the results. We also developed a prototype for semi-automatic rule construction and mapping text to predicate logic².

1.3 Thesis Structure

Chapter 2 discusses the related work. A broad overview of the field is followed by the work that is more related to each of the approaches.

Chapter 3 explains various representations and preliminaries for the proposed approaches. We start with the initial text representations: layered representation, which consists of several layers of annotation and word vector representation, which is a distributed representation. Then, we define textual patterns and their extension, the context-free grammar. Finally, three types of meaning representations are presented: predicate logic, which is used to represent ontologies, our proposed semantic trees and distributed representations, which are part of neural networks, for which we also give a short presentation.

Chapter 4 presents the proposed learning approaches. First, two approaches for mapping text to predicate logic are proposed. One uses textual patterns, while the other is based on syntax parser and ontological resources. Then, we propose an approach for induction of grammar, which is then used to parse text into semantic trees. This is followed by an approach for constructing an ontology from the grammar and semantic trees. The last part of this chapter proposes an approach for constructing NLI hypotheses. In this part we also propose generative, classification and discriminative neural networks that are used for construction and evaluation.

Chapter 5 provides the evaluation of the proposed approaches. We start with a preliminary evaluation of approaches that map to predicate logic. Next, we present the evaluation of the grammar induction approach, followed by the ontology construction approach, both evaluated on the dataset made from Wikipedia person pages. Then, the approach for constructing NLI datasets is evaluated. This includes a comparison of generative models and a comparison of the proposed evaluation metric to other metrics. We conclude this chapter with a qualitative comparison of the approaches and representations, which is backed up by the empirical experimentation.

In Chapter 6 a prototype for semi-automatic creation of rules for mapping text to predicate logic is illustrated.

Chapter 7 provides the conclusion and directions for future work.

¹https://github.com/jstarc/nli_generation

²<https://www.youtube.com/watch?v=a75wqJTW4fM>

Chapter 2

Related Work

Text Understanding, sometimes also referred to as Natural Language Understanding (Allen, 1995) or Machine Reading (Etzioni, Banko, & Cafarella, 2006), is a research effort towards the development of systems that fully automatically understand the meaning of any text they are given. It can also be seen as the ultimate goal of Natural Language Processing (NLP), the term that is associated with a broad range of tasks for automatic text analysis.

Cambria and White (2014) argue that there are three paradigms in NLP. The methods from the syntactic paradigm are based on word and word co-occurrence statistics. Bag-of-words is an import representation of text in this paradigm. These methods usually perform satisfactorily if a large enough body of text is given, and do not work well at the sentence level. The semantic paradigm is concerned with the intrinsic meaning of the text. The methods do not focus on operating with bare words and their statistics, instead they operate with concepts and analyze their relationships. These concepts either exist in an ontology or they are implicitly learned. The semantic paradigm is also the paradigm we most closely follow in our work. For instance, our proposed semantic trees contain nodes that represent ontology concepts. Furthermore, in the NLI dataset generation, the system learns the concepts implicitly, and uses them to generate hypotheses that comply with the label, which defines the semantic relationship between the premise and the hypothesis. Finally, the pragmatic paradigm considers the context of the text. This requires that the systems not only to understand the meaning of the text, but also the intent, circumstances, author, time, location, etc. Historically, most of NLP research has been based on the syntactic paradigm. Recently, a lot of effort has been steered towards the semantic paradigm. While we have not yet witnessed substantial achievements within the pragmatic paradigm, it will be necessary in the future in order to approach complete text understanding.

In this overview of related methods and meaning representations, we start by mentioning NLP tasks that are considered a preprocessing step to the core methods for text understanding. These tasks are either required for our methods, parallel to them, or otherwise important for text understanding. For initial preprocessing, these tasks include word and sentence segmentation. For filtering a particular language or domain tasks, such as language identification or topic recognition modeling, are relevant. Annotating words with various categories is vital for many text understanding methods. The most important tasks for annotation are: named entity recognition, named entity disambiguation, word sense disambiguation, part-of-speech tagging and coreference resolution. Our layered text representation can be constructed using these methods if manual annotations are not provided. A higher order of text annotation is parsing. Many methods use parsers to construct syntax trees or dependency graphs, which are then further exploited by the core methods. Distributed meaning representation learning methods frequently require pretrained word

vectors.

2.1 Meaning Representations

We start our review of meaning representations by presenting representations based on frame semantics (Fillmore, 1977). Frames are characterized by prototypical descriptions of actions, objects, etc. rather than a world model representation. For each frame there are a number of predefined arguments or roles. For instance, a *Sending* frame has roles such as *Sender*, *Recipient* and *Theme* (the object being sent). The goal of the task known as Semantic Role Labeling (SRL) is to select the right frame and associate it to a predicate in a sentence, then assigning words or phrases from a sentence to roles of the frame. FrameNet (Baker, Fillmore, & Lowe, 1998) is a collection of a large number of frames with many of different roles. Another SRL resource is PropBank (Palmer, Gildea, & Kingsbury, 2005), a corpus annotated with predicate argument structures, where predicates are represented by verbs (predicates) and there is a fixed size of their arguments (roles). Many methods approach the problem with supervised learning, where features are constructed from syntactic parse trees (Gildea & Jurafsky, 2002; Xue & Palmer, 2004), while Das and Smith (2011) present a semi-supervised approach to SRL. Furthermore, unsupervised approaches for frame induction have been based on graph partitioning (Lang & Lapata, 2011) and clustering (Titov & Klementiev, 2012). The focus of our work is to extract the meaning from the whole sentence and represent it with a richer set of concepts or a distributed representation.

Abstract Meaning Representation (AMR) (Banarescu et al., 2013) is a generalization of semantic frames. It covers many more language phenomena, such as named entities, co-references, negation, and interrogative sentences. Furthermore, AMR does not model only one frame within a sentence, but represents the whole sentence, where multiple frames are connected. AMR can be represented with three formats: a native AMR format, graph format and logical format. In graph format the AMR can be seen as a rooted, directed, edge-labeled and node-labeled graph. The nodes of the graph are either words or PropBank frames. In the logic format AMR is represented by conjunctions of logical triples with variables. AMR in the graph format resembles our proposed semantic trees. The AMR graph is not limited to a tree and has labeled edges. AMR in logic format resembles our version of predicate logic. The major difference is that the concepts from our representation are grounded in the underlying ontology. On corpora of English sentences paired with AMR representations various machine learning approaches have been applied (Pust, Hermjakob, Knight, Marcu, & May, 2015; Artzi, Lee, & Zettlemoyer, 2015; Flanigan, Thomson, Carbonell, Dyer, & Smith, 2014).

Discourse Representation Structures (DRS), based on Discourse Representation Theory (DRT) (Kamp, 1981), also cover a wide range of linguistic phenomena, including discourse relations, which logically connect different segments of a discourse. Groningen Meaning Bank (Basile, Bos, Evang, & Venhuizen, 2012) is a corpus that is annotated with DRSs. The basic annotation unit is not a sentence, like in most of the corpora, but a discourse (multiple sentences).

Another theoretically well grounded meaning representations are first-order logic (FOL) and lambda calculus. Lambda dependency-based compositional semantics (lambda DCS) (Liang, Jordan, & Klein, 2013) is a more compact version of lambda calculus, where variables are eliminated and existential quantification is made implicit.

Some meaning representations have emerged with the Semantic Web. Resource Description Framework (RDF) offers a representation used for conceptual description of web resources. The representation is based on subject-predicate-object relations, therefore it

can also be used to represent relations extracted from text. Furthermore, reasoning engines have been developed to check the semantic consistency of RDF triples. RDF was extended by the Web Ontology Language (OWL)(McGuinness & Van Harmelen, 2004), which is based on description logic and provides a more suitable infrastructure for representing ontologies.

2.1.1 Existing ontologies

Many structured collections of knowledge can be exploited for representing the meaning or in the learning process. These collections come in various forms: as dictionaries, databases, knowledge bases, taxonomies and ontologies. Our work is mostly focused on ontologies. They are the most general knowledge representation, thus also other mentioned collections can be represented as ontologies.

In Section 3.3.1.1 we present three general domain ontologies that we have used in our work: Cyc (Lenat, 1995), a common sense ontology, DBpedia (Lehmann et al., 2015), an ontology extracted from Wikipedia, and Freebase (Bollacker, Evans, Paritosh, Sturge, & Taylor, 2008), a collaboratively constructed ontology. Other most prominent ontologies for text understanding include WordNet (Fellbaum, 1998), a lexical database of English language, ConceptNet (H. Liu & Singh, 2004), a common sense knowledge base Yago (Suchanek, Kasneci, & Weikum, 2007), a knowledge base automatically extracted from Wikipedia, WikiData (Vrandečić & Krötzsch, 2014), a collaboratively edited knowledge base, NELL (Carlson et al., 2010), a knowledge base that is constructed using a never-ending language learning system that reads the Web, and BabelNet (Navigli & Ponzetto, 2010), a multilingual ontology.

2.2 Learning Approaches

In this section, we discuss the tasks and approaches that are related to our proposed approaches in Chapter 4.

2.2.1 Semantic parsing

The process of mapping a text sentence into a meaning representation is also known as semantic parsing. It can also be seen as the reverse task of language generation. In the beginning semantic parsers were mostly hand-crafted (Warren, 1981). It turned out that developing training corpora is not more difficult than manually designing robust semantic parsers. With the availability of training corpora and improved hardware, learning approaches started to emerge. One of the prominent early datasets are Geoquery (Zelle & Mooney, 1996) and Atis (Dahl et al., 1994), which map sentences to domain-specific databases.

This has allowed the development of supervised approaches that learn from sentence – meaning representation pairs. Several approaches have used Combinatory Categorical Grammar (CCG) and lambda calculus as a meaning representation (Kwiatkowski, Zettlemoyer, Goldwater, & Steedman, 2011; Krishnamurthy & Mitchell, 2015; Reddy, Lapata, & Steedman, 2014). CCG grammar closely connects syntax and semantics with hand-crafted lexicon, where each entry consists of a term, a syntactic category and a lambda statement. Similarly, our proposed context-free grammar contains production rules. Some of these rules do not contain lexical tokens (the grammar is not lexicalized), which gives the ability to express some relations with a single rule. For instance, to parse jazz drummer, rule $\langle \text{Musician Type} \rangle ::= \langle \text{Musical Genre} \rangle \langle \text{Musician Type} \rangle$ is used to directly express the

relation, which determines the genre of the musician. Ge and Mooney (2005) have developed an approach that semantically augments syntactic parse trees. Furthermore, Y. W. Wong and Mooney (2006) have developed a statistical machine learning approach, which eliminates the requirement of a hand-crafted lexicon. A generative approach, which generates trees containing both lexical words and meaning representation, was developed by Lu, Ng, Lee, and Zettlemoyer (2008). The approach does not learn an explicit grammar, as we do in our work, but rather learns to recursively create nodes at each level according to a Markov process.

Later on datasets for question answering based on Freebase were created – Free917 (Cai & Yates, 2013) and WebQuestions (Berant, Chou, Frostig, & Liang, 2013). These datasets contain short questions from multiple domains, and since the meaning representations consist of Freebase concepts, they allow reasoning over Freebase’s ontology, which is much richer than databases in GeoQuery and Atis. The WebQuestions dataset does not contain the full meaning representation, but only the entity, which is the answer to the question. Therefore, WebQuestions mandates weakly supervised learning, where the meaning representation is first formed, then it is executed on the ontology as a query to get the answer (Liang et al., 2013). Another approach learns low-dimensional dimensional embeddings of words and ontology concepts, which is used to score questions against candidate answers (Bordes, Chopra, & Weston, 2014).

Many datasets were constructed by either forming sentences given the meaning representation or vice-versa. Consequently, systems that were trained and evaluated on these datasets may not work on sentences that cannot be represented by the underlying ontology. To overcome this limitation Krishnamurthy and Mitchell (2015) developed an open vocabulary semantic parser. Their approach uses a CCG parser on questions to form lambda statements, which besides Freebase vocabulary, contain underspecified predicates. These lambda statements are together with answers – Freebase entities – used to learn a low-dimensional probabilistic database, which is then used to answer fill-in-the-blank natural language questions. In a similar fashion, Choi, Kwiatkowski, and Zettlemoyer (2015) define underspecified entities, types and relations, when the corresponding concept does not exist in Freebase. Our work on semantic trees also uses underspecified concepts in form of null nodes. From these nodes our approach learns new grammar rules or new instances.

2.2.2 Ontology learning

In this dissertation, we propose an approach for ontology learning from grammar and semantic trees. Many other ontology learning approaches address the same ontology components as in our approach. However, their goal is to learn only the salient concepts for a particular domain, while our goal is to learn all the concepts (including instances, like particular organizations), so that they can be used in the meaning representation. As survey by W. Wong, Liu, and Bennamoun (2012) summarizes, the learning mechanisms are based either on statistics, linguistics, or logic. Our approach is unique because part of our ontology is constructed from the grammar. Many approaches use lexico-syntactic patterns for ontology learning. These are often based on dependency parses, like in Zouaq and Nkambou (2009) and Völker, Hitzler, and Cimiano (2007). Our approach does not rely on pretrained parsers, which makes it suitable for non-standard texts and poorly resourced languages. Our approach also builds patterns, however in the form of grammar rules. Instead of lexico-syntactic patterns, which contain linguistic classes, our approach models semantic patterns, which contain semantic classes, like *Person* or *Organization*. These patterns are constructed in advance, which is sometimes difficult because the constructor is not always aware of all the phenomena that are expressed in the input text. Our approach allows to create a small number of seed patterns in advance, then explore

other patterns through the process of grammar learning. A similar bootstrapping semi-automatic approach to ontology learning was developed by W. Liu, Weichselbraun, Scharl, and Chang (2005), where the user validates lexicalizations of a particular relation to learn new instances, and by Brewster, Ciravegna, and Wilks (2002), where the user validates newly identified terms, while in our approach the user validates grammar rules to learn the composition of whole sentences. A similar approach with combining DBpedia with Wikipedia for supervised learning has been taken in Walter, Unger, and Cimiano (2014), however their focus is more on learning lexicalizations of relations and classes.

Many approaches focus only on one aspect of knowledge extraction, like taxonomy extraction (Navigli, Velardi, & Faralli, 2011; Snow, Jurafsky, & Ng, 2006) or relation extraction (Bunescu & Mooney, 2005; Zelenko, Aone, & Richardella, 2003). Combining these approaches can lead to cumbersome concept matching problems. This problem was also observed by Poon and Domingos (2010). Their system OntoUSP tries to overcome this by unsupervised inducing and populating a probabilistic grammar to solve the question answering problem. However, the result are logical-form clusters connected in an *isa* hierarchy, not grounded concepts, which are connected with an existing ontology. Another bootstrapping semi-automatic approach was developed for knowledge base population (Wolfe et al., 2015). The task of knowledge base population is concerned only with extracting instances and relations given the ontology. In our work, we also extract the backbone of the ontology – classes and taxonomic relations.

2.2.3 Grammar induction

We propose a semi-automatic method that induces a grammar suitable for the scenario, in which an ontology is extracted, and text is parsed into semantic trees. A survey by D’Ullizia, Ferri, and Grifoni (2011) compares several papers on grammar induction. According to their classification, our method falls into unsupervised, text-based (no negative examples of sentences) methods. Many such methods induce context-free grammars. However, their focus is more on learning syntactic structures rather than semantic. This is evident in evaluation strategies, where their parse trees are compared against golden parse trees in treebanks, like Penn treebank (Marcus, Marcinkiewicz, & Santorini, 1993), which are annotated according to syntactic policies. Furthermore, our grammar is not limited to a specific form, like for instance Chomsky normal form or Greibach normal form, instead it may contain arbitrary context-free rules. Several algorithms, like ours, employ the greedy strategy of grammar induction, where the grammar is updated with the best decision at each step. Whereas our method adds a rule after all sentences are parsed, the Incremental Parsing Algorithm (Seginer, 2007) updates the grammar after each sentence. This is also done in the ADIOS method (Solan, Horn, Ruppel, & Edelman, 2005), where it has been shown that the order of sentences affects the grammar. Our method employs a frequency analysis and human supervision to control the grammar construction, while others use the Minimum Description Length principle (Watkinson & Manandhar, 2001), clustering of sequences (Clark, 2001), or significance of word co-occurrences (Hänig, Bordag, & Quasthoff, 2008).

2.2.4 Natural Language Inference

The goal of Natural Language Inference (NLI) is to correctly determine whether a sentence (referred to as a hypothesis) entails or contradicts or is neutral in respect to another sentence (a premise). It has also been the focal point of Recognizing Textual Entailment (RTE) Challenges (Dagan, Glickman, & Magnini, 2006). The approaches to NLI include the bag-of-words matching approach (Glickman, Dagan, & Koppel, 2005), the matching

predicate argument structure approach (MacCartney, Grenager, de Marneffe, Cer, & Manning, 2006) and the logical inference approach (Bos & Markert, 2006). More recently, the Stanford Natural Language Inference (SNLI) dataset has been collected (Bowman, Angeli, et al., 2015). Having over a magnitude more examples than previous corpora, it has spawned the development of neural network approaches (Rocktäschel et al., 2016; Wang & Jiang, 2016; Cheng et al., 2016; Parikh et al., 2016). The current state-of-the-art model for classification (Parikh et al., 2016) achieves 86.6% accuracy on the SNLI dataset. Instead of classification our work focuses on the generation of NLI datasets. A generation approach was also proposed by Kolesnyk, Rocktäschel, and Riedel (2016), whose focus was on generating entailment inference chains.

2.2.5 Natural Language Generation

Natural Language Generation (NLG) is the task of generating a natural language from a structured form, such as knowledge base or logic form (Wen et al., 2015; Mairesse et al., 2010; Belz, 2008). The input in our task is unstructured text (premise) and label. On the other side of this spectrum, there are tasks that deal solely with unstructured text, like machine translation (Koehn, 2009; Bahdanau, Cho, & Bengio, 2014; Luong, Sutskever, Le, Vinyals, & Zaremba, 2015), summarization (Clarke & Lapata, 2008; Rush, Chopra, & Weston, 2015) and conversational dialogue systems (Serban, Sordoni, Bengio, et al., 2016; Banchs & Li, 2012). Another recently popular task is generating captions from images (Vinyals, Toshev, Bengio, & Erhan, 2015; Socher, Karpathy, Le, Manning, & Ng, 2014). In our work, we propose a metric for evaluation of text-generating models. As recognized by Reiter and Belz (2009), the metrics of text-generating models fall into three categories: manual evaluation, automatic evaluation metrics, task-based evaluation. In the evaluation based on human judgment each generated textual example is inspected manually. The automatic evaluation metrics, like ROUGE, BLEU and METEOR, compare human texts and generated texts. Elliott and Keller (2014) show that METEOR has the strongest correlation with human judgments in the image description evaluation. The last category is the task-based evaluation, where the impact of the generated texts on a particular task is measured. This type of evaluation usually involves costly and lengthy human involvement, like measuring the effectiveness of smoking-cessation letters (Reiter, Robertson, & Osman, 2003). On the other hand, the task in our evaluation, the NLI classification, is automatically done. In Hodosh, Young, and Hockenmaier (2013) ranking was used as an automatic task-based evaluation for associating images with captions.

2.2.6 Neural networks for NLP

With the advancement of deep learning, many neural network approaches have been proposed for tackling NLP problems. Neural networks have been frequently used as a non-linear substitute for classification tasks. For instance, fully-connected architectures have been used for dependency parsing (Chen & Manning, 2014), sentiment analysis, factoid question answering (Iyyer, Manjunatha, Boyd-Graber, & Daumé III, 2015), etc. Convolutional neural networks have been used for document classification (Johnson & Zhang, 2014), semantic role labeling (Collobert & Weston, 2008), NLI (Mou et al., 2016), etc. Recursive neural networks have been used for constituency (Socher, Bauer, Manning, & Ng, 2013) and dependency (Le & Zuidema, 2014) parsing, question answering (Iyyer, Boyd-Graber, Claudino, Socher, & Daumé III, 2014), etc. Finally, the Recurrent Neural Networks (RNNs), which have also been used in our work, have been used for sequence tagging (Irsoy & Cardie, 2014), normalizing tweets (Chrupała, 2014), etc. RNNs are frequently designed with attention and memory mechanisms (Luong, Pham, & Manning,

2015; Sukhbaatar, Weston, & Fergus, 2015), who allows the network to refer back to the input sequence.

Besides other tasks RNNs are also suitable for text generation. The Recurrent Neural Network Language Model (RNNLM) (Mikolov, Karafiát, Burget, Cernocký, & Khudanpur, 2010) is one of the simplest neural architectures for generating text. The approach was extended by (Sutskever et al., 2014), which use an encoder-decoder architecture to generate a sequence from the input sequence. The Hierarchical Recurrent Encoder-Decoder (HRED) architecture (Serban, Sordoni, Bengio, et al., 2016) generates sequences from several input sequences. These models offer very little variety of output sequences. It is obtained by modeling the output distribution of the language model. To introduce more variety, models based on variational autoencoder (VAE) (Kingma & Welling, 2013) have been proposed. These models use stochastic random variables as a source of variety. In Bowman, Vilnis, et al. (2015) a latent variable is used to initialize the RNN that generates sentences, while the Variational Recurrent Neural Network (VRNN) (Chung et al., 2015) models the dependencies between latent variables across subsequent steps of RNN. The Latent Variable Hierarchical Recurrent Encoder-Decoder (VHRED) (Serban, Sordoni, Lowe, et al., 2016) extends the HRED by incorporating latent variables, which are learned similarly than in VAE. The latent variables are, like in some of our models, used to represent the mappings between sequences. Conditional variational autoencoders (CVAEs) (Yan, Yang, Sohn, & Lee, 2015) were used to generate images from continuous visual attributes. These attributes are conditional information that is fed to the models, like the discrete label is in our models.

Chapter 3

Representations

3.1 Text Representations

In this section, we present two representations of text that we use throughout this dissertation: the layered representation and word vector representation. We start with a sentence as a basic unit for text understanding in our work. Longer sequences of ordered sentences, such as paragraphs and articles, are split, thus the order of the sentences and the information spread across the sentences is not considered. Then, if a sentence is not already split into words, a word tokenizer is used. From here we distinguish between two different representations of words. In the first representation each word is represented by several layers. Each layer has its own linguistic or semantic interpretation. In the second representation each word is represented by a vector.

3.1.1 Layered representation

Each sentence is represented by several layers. Each layer consists of several tokens, which span over one or more words. The basic layer is the lexical layer, where each token represents a single word. An example is presented in Table 3.1. Other layers are created from annotations such as named entities, part-of-speech tags, concepts from ontology, etc. The annotations are manually created, alternatively they can be obtained by shallow NLP tools, such as named entity recognizer, part-of-speech tagger, etc. Some annotations, like named entities, may span over several words. The words that do not have an annotation are represented by a null token. A *term* is a subpart of the sentence defined by the starting and ending word position in the sentence. It has a different interpretation in each layer. If the interpretation breaks any of the annotations, it is not valid. For instance, a term representing Madeira is not valid in *named-entity* layer in Table 1 because it breaks the Person token. Layered representation allows us to build more general structures or patterns, which contain terms from different layers.

3.1.2 Word vector representation

In the word vector representation, also called word embeddings or distributed word representation, each unique word is represented by a fixed-length vector. Each word is represented by a point in the vector space model. The model is trained based on *distributional hypothesis*, which states that words with similar semantics occur in similar contexts. Thus, semantically similar words appear close to each other in the space model. The representation is a dense representation suitable for the input to deep neural networks.

Table 3.1: Layered representation of a sentence "Phil Madeira is a musician from Nashville." Null tokens are expressed as "-".

Layer	Tokens						
lexical	Phil	Madeira	is	a	musician	from	Nashville
lower-case	phil	madeira	is	a	musician	from	nashville
named-entity	Person		-	-	-	-	Location
instance	Phil_Madeira		-	-	Musical_Artist	-	Nashville_Tennessee
class	Person		-	-	Profession	-	Location

3.2 Textual Patterns

One approach to text understanding is to search for parts of text that are similar to each other and preferably have a common meaning. These parts may be represented by a single pattern, which consists of consecutive tokens in the layered representation. The tokens can be from different layers. For example, *[Person] is a [Profession]* is a pattern from the example in Table 3.1, where *Person* is a token from named-entity layer and *Profession* is a token from the class layer. This pattern *covers* term *Phil Madeira is a musician*, but it can also cover a term like *Bob is a lawyer*. *Person* and *Profession* are the arguments of this pattern, thus they are surrounded with brackets. The arguments are more general parts of the pattern coming from a more generic layer such as named-entity layer.

The pattern from the previous example only covers one part of the sentence. To cover a larger part of the sentence, one might create a longer pattern such as *[Person] is a [Profession] from [Location]*. There are two issues present here. First, the pattern is more specific, thus it may not cover as much in the dataset. Second, the pattern is not atomic from the meaning standpoint because the information about profession and location can be separated. One solution is to apply several patterns and then combine them together in a recursive way. For instance, *[Person] is a <Anything>* and *[Profession] from [Location]*. The *<Anything>* is a universal token that can cover any part of text. The problem is that when a universal token is matched against the text, there are multiple possible matches, from a single word (after word "a"), up to all the remaining words in the sentence. The next section presents how to utilize a context-free grammar to apply several patterns recursively. In this case, free tokens cover all the remaining tokens in the sentence. Another solution is not allowing universal tokens to appear at the beginning or at the end of patterns. We name the remaining patterns – *gap* patterns. For instance, to match the universal token *<Position>* in pattern *[Person], <Position> of [Organization]*, all tokens between "," and "of" are taken.

3.2.1 Grammar approach

In this section, we will extend and formalize that approach by defining a context-free grammar approach. The main component on this grammar are production rules such as *<Relation> ::= <Person> is <LifeRole>*. These rules are together with the text in layered representation processed by a recursive descent parser with backtracking. A *semantic tree* is generated as a result. An example is presented in Figure 3.1 and more details are given in Section 3.3.2.

The proposed context-free grammar G is defined by the 5-tuple: $G = (V, \sigma, P, S, R)$, where

- V is a set of *non-terminals*. Each non-terminal represents a semantic class, for example $\langle \text{Person} \rangle$, $\langle \text{Location} \rangle$, $\langle \text{Organization} \rangle$. There is also a universal non-terminal $\langle * \rangle$, which can be replaced by any other non-terminal. The replacing non-terminal replaces all occurrences in a rule. It is used to represent several rules with a single notation. The grammar with universal non-terminals is still context-free. For more examples, see *seed rule examples* in Section 4.3.4.
- σ is a set of *terminals*. Terminal is any existing non-null token from any sentence layer. We denote a terminal by *value*{*layer*}. For instance, [location]{named-entity}, Phil_Madeira{instance}, etc. If the terminal is from the lexical layer or the layer information is not relevant, the layer is skipped in the notation.
- P is a set of production rules that represents a relation from $V \rightarrow (V \cup \sigma)^*$. For example, $\langle \text{Relation} \rangle ::= \langle \text{Person} \rangle$ is $\langle \text{Life Role} \rangle$
- S is the root non-terminal symbol. Non-terminals represent semantic classes, the starting symbol is thus chosen based on the semantic class of the input examples. If the input examples are sentences, then the appropriate category is $\langle \text{Relation} \rangle$. Whereas if the input examples are noun phrases, the starting symbol may be a more specific category, like $\langle \text{Job Title} \rangle$.
- R is a set of properties: $\{positive, neutral, negative, non-inducible\}$. The property controls the usage of the rule in the parsing and rule induction step. More details are given in Section 4.3.3.

3.3 Meaning Representations

3.3.1 Predicate logic

In this section, we define a predicate logic language that closely resembles CycL(Guha & Lenat, 1991), the language of Cyc. This language is used to define ontologies in this dissertation. The meaning is represented by *concepts*, which form *relations*. The most basic form of relations are predicate relations, where *predicate* is in the first place of a relation, followed by one or more arguments. Predicate is a truth function that determines the relationship between concepts. For instance, $(father\ John\ Bob)$ is a relation that represents the meaning of a sentence "John is the father of Bob." The *father* is the predicate, which characterizes the relationship between two arguments – John and Bob in this example. All concepts together form an *ontology*, and can be of the following types:

- *Instances* represent individual concepts, like *BarackObama*, *Paris*, *Red-Color*, etc.
- *Classes* represent a collection of instances with a particular characteristic, like *Person*, *Organization*, *Profession*, etc.
- *Literals* represent strings, numbers, characters, etc. For instance, "London", 18.1, "w", etc.
- *Predicates* define the type of relationship between concepts, for instance *grandmother*, *capitalCityOf*, *dateOfEvent*, etc. Furthermore, we highlight two taxonomic predicates. The *isa* predicate relates an instance to a class, whose member it is. The *subClassOf* predicates relate a class to a class that subsumes it. Predicates are not

necessarily placed on the first place in the relation. For instance, when expressing facts about predicates, like *(argIsa father 1 Animal)*, which states that the first argument of the father relation must be an instance of class Animal.

- *Functions* are used to form compound concepts. For instance *(FatherFn Bob)* relation represents the father of Bob. This concept is equivalent to the concept of John. Functions are useful to express concepts, without explicitly creating them. For instance, *ProfessorFn* is used to express a class of professors that teach a particular subject, like *(ProfessorFn Math)*. This function allows us to create a professor class of any subject, even if it does not yet exist in the ontology.
- *Relations* express relationships between concepts. They can also be used as an argument of another relation. For example, *(doubts Alice (age John 53))*.
- *Logical Constants* are predicates, such as *and*, *or*, *implies*, *thereExists*, that connect relations. The following example uses logical constants to express an event, where John is running:

```
(thereExists ?EVENT
  (and
    (isa ?EVENT Running)
    (doneBy ?EVENT John)))
```

Terms that start with a question mark symbolize variables, like ?EVENT in the previous example.

- *Rules* are used in inference for deriving new conclusions. For instance,

```
(implies
  (father ?CHILD ?FATHER)
  (child ?FATHER ?CHILD))
```

is used to derive *(child Bob John)* from *(father John Bob)*. Rules are also utilized to reject relations inconsistent with the ontology. For instance, *(child Bob NewYorkCity)* is inconsistent with *(argIsa child 2 Animal)*, which states that the second argument of *child* relation is a type of animal.

- *Queries* are utilized to answer questions about the knowledge in the ontology. They are structured like relations, however they can contain variables. For example, *(father John ?PERSON)* is a query equivalent to "Who is the father of John?". The answer are all concepts that replace the variables to form relations that are in the ontology. In our example, *Bob* is the answer to the query if *(father John Bob)* is in the ontology.

In this dissertation, the following naming is used: Predicates start with a small letter, while other concepts start with a capital letter. Functions end with a *Fn* suffix.

From the text understanding standpoint, the goal is to encode the meaning of text in the proposed language and add this encoded representation to the ontology. Once it is available in the ontology, it can be utilized for various applications that involve querying, reasoning, etc.

Some concepts are present in many relations and rules, thus they are frequently featured in the inference and may help produce useful deducted relations, contradictions, or query results. We refer to these concepts as *semantically rich concepts*. For example, if concept *Amazon* is not connected to any of the other concepts in the ontology, then we do not even know if the concept represents the company, the river or something else. Providing this information will make the concept semantically richer. On the other hand, the *semantically poor concepts* are not connected with many other concepts. We also refer to a concept as

semantically poor if they are correct but not optimal mapping for a particular text. For instance, to express “son of” relationship predicate *son* is more optimal than predicate *child*. In this case, using predicate *child* is not incorrect since the relation is still true, however the predicate *child* is semantically poorer.

3.3.1.1 Existing ontologies

In this section, we present existing ontologies that are used in our work. The ontologies are expressed with the predicate logic language presented in the previous section. These ontologies are also accompanied with an infrastructure that allows their manipulation, such as querying, adding new knowledge, etc. We present the following ontologies:

- *DBpedia* (Lehmann et al., 2015; “DBpedia,” 2016) is an ontology that is extracted from Wikipedia. Every instance in DBpedia represents a page in Wikipedia. There is less need for word sense disambiguation when using Wikipedia as a textual dataset because of the links that point to other pages – DBpedia instances. DBpedia has around 5 million entities and a few billion triples. The triples are relations with two atomic arguments. They are represented in the Resource Description Framework (RDF) format. DBpedia also contains many links to other RDF ontologies.
- *Freebase* (Bollacker et al., 2008; “Freebase,” 2016) was a collaborative ontology meaning that community members add and edit the data. The data is harvested from many sources including DBpedia, thus many instances link to their corresponding DBpedia instances. Since Freebase has around 50 million entities, there are also many instances that are not represented in DBpedia. Freebase is also represented with RDF triples and contains a few billion of them.
- *Cyc* (Lenat, 1995; “Cyc,” 2016) is a project that attempts to build an ontology that will allow human-like reasoning. Compared to DBpedia and Freebase, Cyc’s ontology contains more everyday common sense and abstract knowledge. The ontology contains a few hundred thousands concepts and few a million relations, which is fewer than the other two ontologies. However, Cyc’s ontology also contains many rules, which are used to implicitly derive new relations and reject contradictory relations. The language of Cyc, *CycL*, is more expressive than triples, and our proposed language based on it. Cyc’s ontology is accompanied with the inference engine that allows reasoning (explaining why is something true or false), natural language generation, etc. Our work was done on ResearchCyc, a version that is available for research purposes. There is also OpenCyc, a more limited version but unrestricted for usage. Part of it is also available in the RDF format.

3.3.2 Semantic trees

Semantic trees represent meaning. They are obtained by parsing some text with a context-free grammar defined in Section 3.2.1. An example of a semantic tree is depicted in Figure 3.1. Semantic trees consist of several nodes, which are connected into a tree structure that models the meaning of the sentence. Each node consists of a class, a rule and a term. These are displayed from top to bottom in each node in Figure 3.1. Terms are sections of the sentence and are interpreted in the lexical layer in Figure 3.1. The class of the node is the semantic class of the term and is represented by a single non-terminal. This non-terminal is usually the left-hand side of the rule, however this is not the case if the left-hand side is a universal non-terminal as in the right node in the second level in Figure 3.1.

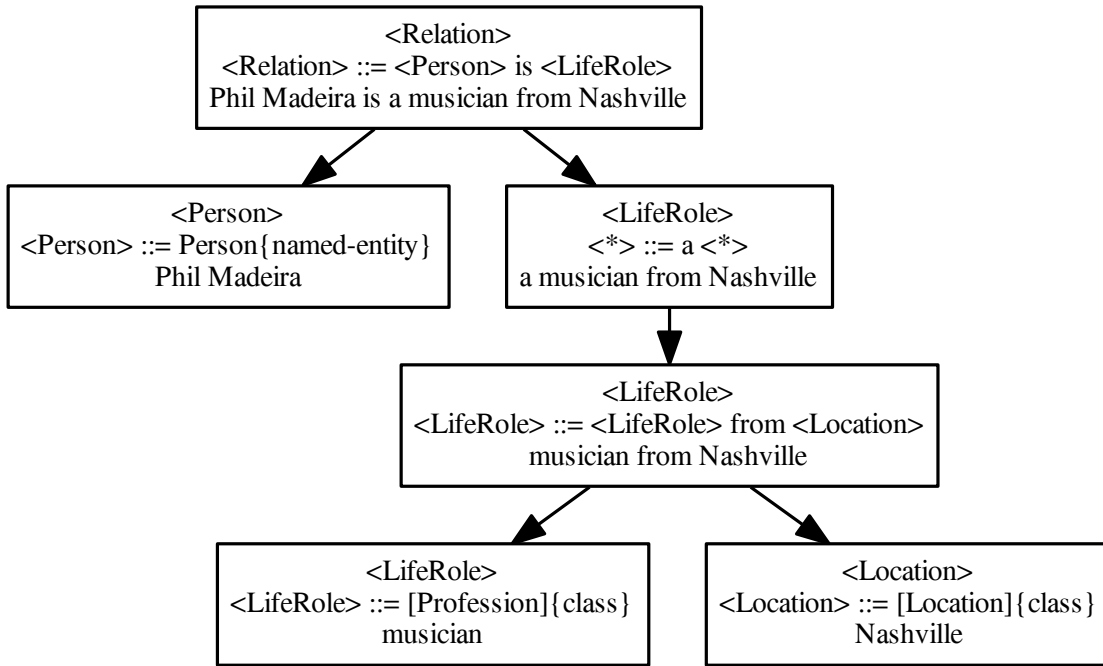


Figure 3.1: An example of a semantic tree, which is the result of parsing the input from Table 3.1. The tree represents the sentence “Phil Madeira is a musician from Nashville”. The tree is fully parsed. Each node has three rows: the class, the rule and the term interpreted in the lexical layer.

Contrary to dependency and constituent parse trees, some words in semantic trees do not have their own node. In fact, some words are grouped together in leaf nodes to represent atomic instances, like “Phil Madeira” in our example. Furthermore, some words only appear in terms of inner nodes.

3.3.3 Distributed representation

Previously, we presented representations based on local representation schema, where each concept is represented by a unique atomic representation unit. In the distributed representation each concept is represented by many representation units, and each representation unit is involved in the representation of each concept. While the concepts in our local representations are defined with a symbolic language, the concepts in distributed representations are represented by vectors, matrices, tensors, etc. An example of distributed representations are word vectors presented in Section 3.1.2. The dimensionality (the number of representation units) in distributed representation is fixed. Thus, adding a new concept does not increase the dimensionality, as in the local representation. Furthermore, in a local representation the concepts are represented independently. Without an ontology that connects them, they are all equally different from each other. In the distributed representation the learning models can respond to the similarity between the unseen example and examples it has been trained on. The distributed representations are suitable for neural networks, where they can represent inputs, outputs, or internal states.

3.3.3.1 Neural networks

Neural networks, also known as Artificial Neural Networks (ANNs), are a family of parametric learning models used to approximate functions. The input to the networks in our work is a sequence or several sequences of word vectors. The output depends on the learning problem. The output of classification models is a vector representing probabilities of classes, the output of generative models is a sequence of vectors representing probabilities of words, etc. The goal of all models is to learn the parameters, which minimize the loss function defined for the model. The models extract the meaning from the textual input needed to construct the desired output.

Recurrent Neural Networks (RNNs) are neural networks very suitable for processing sequences. They are called recurrent because they process each element of the sequence in the same way, and the output at each step is dependent on the previous computations. They are the basic building block in our neural networks. We use two variants of RNNs – Long Short Term Memory (LSTM) network (Hochreiter & Schmidhuber, 1997) and an attention-based extension of LSTM, the mLSTM (Wang & Jiang, 2016). The LSTM tends to learn long-term dependencies better than vanilla RNNs. The input to the LSTM is a sequence of word vectors $X = x_1x_2 \dots x_n$, and the output is a sequence of vectors $H = h_1h_2 \dots h_n$. At each time point t , input gate i_t , forget gate f_t , output gate o_t , cell state C_t and one output vector h_t are calculated.

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (3.1)$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (3.2)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (3.3)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tanh(W_c x_t + U_c h_{t-1} + b_c) \quad (3.4)$$

$$h_t = o_t \odot \tanh(C_t), \quad (3.5)$$

where σ is a sigmoid function, $\odot$ is the element-wise multiplication operator, $W \in \mathbb{R}^{d \times e}$ and $U \in \mathbb{R}^{d \times d}$ are parameter matrices, $b \in \mathbb{R}^d$ parameter vectors, e is the input vector dimension, and d is the output vector dimension. The vectors C_0 and h_0 are set to zero in the standard setting, however, in some cases in our models, they are set to a value that is the result of previous layers.

The mLSTM is an attention-based model with two input sequences – premise and hypothesis in case of NLI. Each word of the premise is matched against each word of the hypothesis to find the soft alignment between the sentences. The mLSTM is based on LSTM in such a way that it remembers the important matches and forgets the less important. The input to the LSTM inside the mLSTM at each time step is $x'_t = [a_t, x_t^h]$, where a_t is an attention vector that represents the weighted sum of premise sequence, where the weights present the degree to which each token of the premise is aligned with the t -th token of the hypothesis x_t^h , and $[*, *]$ is the concatenation operator. More details about mLSTM are presented in (Wang & Jiang, 2016).

Chapter 4

Learning Approaches

This chapter presents our proposed approaches to Text Understanding using the representations and preliminaries described in Chapter 3.

We start by proposing two rule-based approaches for mapping text to predicate logic. A textual pattern approach (Section 4.1) and syntax parser based approach (Section 4.2) are presented. The grammar induction approach proposed in Section 4.3 is a generalization of the textual pattern approach. Using the induced grammar, a taxonomy is constructed (Section 4.4). The grammar is also used to parse sentences into semantic trees. These are then utilized to construct predicate logic relations (Section 4.5.2). The last proposed approach is based on the distributed representations and neural networks to construct NLI datasets (Section 4.6). The main contributions are a neural network model for generating NLI hypotheses and the proposed metrics for evaluation of hypothesis. This approach including its evaluation (Section 5.4) is also presented in Starc and Mladenić (2016).

4.1 Mapping to Predicate Logic with Textual Patterns

One approach for mapping text to predicate logic is to define a set of rules, which are applied on the text. The result of the process are predicate logic relations that reflect the meaning of the text. A rule consists of a textual pattern on the left-hand side and a logic pattern on the right-hand side. The main contribution of this section is the procedure for finding valuable textual patterns.

Textual patterns, presented in Section 3.2, consist of fixed tokens and arguments. Logic patterns are predicate logic relations with free variables. The number of arguments in the textual pattern must match the number of free variables. This is an example of a mapping rule

$$[\text{Person}] \text{ is a } [\text{Profession}] \implies (\text{profession ?ARG1 ?ARG2})$$

The mapping algorithm works in a recursive fashion. It searches for a rule that matches the input text. Once the match is found, the terms that match the arguments of the textual pattern are extracted. For each extracted term the algorithm tries to match all the rules again. The output of the rule application is a logic relation, which consists of the logic pattern, where the free variables have been replaced by logic relations or concepts from the rule application of arguments. For example, the following mapping was constructed using the rule from the previous example

$$\text{Phil Madeira is a musician} \implies (\text{profession PhilMadeira Musician})$$

Some of the rules may be generated implicitly. For instance, the rules that do not have any arguments can be generated using the ontology. For instance, if relation (nameString PhilMadeira “Phil Maderira”) exists in the ontology, rule $Phil\ Madeira \implies PhilMadeira$ can be generated. The proposed approach assumes that concepts in the logic pattern already exist in the ontology.

4.1.1 Creating rules

To construct a mapping rule one must create a textual pattern and a logical pattern. The logical pattern is constructed manually, while the textual pattern is constructed semi-automatically using layered generalization of text and a method for discovering promising patterns.

4.1.1.1 Layered generalization

The layered generalization is done by selecting tokens from different layers of the layered representation (see Section 3.1.1 to obtain a pattern representing a more general meaning of the text). The output of this process is a sequence of tokens, which might be from different layers. For each position in the term a single layer is selected, according to predefined layer order. In the beginning, the term is generalized with the first layer. All the non-null tokens from this layer are taken to be part of the generalized term. All the positions of the term that have not yet been generalized are attempted to be generalized with the next layer, etc. The last layer is without null-tokens, therefore each position of the term is assigned a layer. Usually, this is the lexical layer. For example, top part of Table 4.1 shows a generalization of the term from Table 3.1. The layer list is constructed manually. Good layers for generalization are typically those that express semantic classes of individual terms. Preferably, these types should not be too general (loss of information) nor too specific (too many rules).

4.1.1.2 Finding valuable textual patterns

In order to construct rules that will yield good logic representations of meaning, we present an approach, which allows one to find promising textual patterns. First, the text in the layered representation is layer generalized. The tokens from particular (more generic) layers, like named-entity layer, will present arguments of the textual patterns. From each sentence all possible n-grams up to a certain length are extracted. For each unique n-gram the following statistics are calculated

- **Frequency (Fq)** – the number of times the n-gram appears in the whole corpus.
- **Arguments (Args)** – the number of generalized tokens in the n-gram.
- **Stop words (StopW)** – the number of stop words (tokens) in the n-gram including all non-alphabetical tokens.
- **Minimal Token Frequency (MinTokFq)** – token frequency presents the number of times a token appears in the corpus. MinTokFq presents the minimum token frequency out of all the non-argument tokens in the n-gram.
- **Normalized Expectation (NExp)** is the average expectation of one word occurring in a given position knowing the presence of other n-1 words also constrained by

Table 4.1: Two step generalization of the term from Table 3.1. The example proceeds from the top (Original sentence) to the bottom (Bottom-up parsing). The last row is the result of the generalization. In the first step, the layered generalization, the layers were applied in the reverse order as in Table 3.1. Therefore, first the *class* layer was applied, then *instance* and *named-entity* layers have not yielded any new tokens. Finally, *lower-case* layer was used to add the remaining tokens. In the bottom-up parsing step, suppose there are two rules in the grammar – $\langle \text{Life Role} \rangle ::= \text{Profession from Location}$ and $\langle \text{Life Role} \rangle ::= a \langle \text{Life Role} \rangle$

Original sentence					
Phil Madeira	is	a	musician	from	Nashville

Layered generalization					
Person class	is lower-case	a lower-case	Profession class	from lower-case	Location class

Bottom-up parsing			
Person class	is lower-case	a lower-case	$\langle \text{Life Role} \rangle$ non-terminal
Person class	is lower-case	$\langle \text{Life Role} \rangle$ non-terminal	

their positions.

$$Nexp([w_1 w_2 \dots w_n]) = \frac{p([w_1 w_2 \dots w_n])}{\frac{1}{n} \sum_{i=1}^n p([w_1 w_2 \dots \hat{w}_i \dots w_n])}$$

where $p([w_1 w_2 \dots w_n])$ denotes the probability of n-gram $[w_1 w_2 \dots w_n]$ occurring in the corpus. Term $\hat{w}_i$ signifies that word w_i was omitted from the n-gram. It was shown in Dias, Guillore, and Lopes (1999) that normalized expectation together multiplied by frequency is a good measure for finding multi-word units.

These statistics help highlight promising textual patterns, or exclude the less promising ones. Similar statistics – frequency and reliability measure – have been incorporated for selecting the best rule in the grammar induction approach in Section 4.3.

4.2 Mapping to Predicate Logic Using the Syntax Parser

We propose an approach for constructing predicate logic relations utilizing a syntax parser. The main contribution is a translator, which utilizes the proposed high-level abstract rules to connect the output of a syntax parser to logical patterns. The approach is based on XLE parser (Maxwell & Kaplan, 1996) and Cyc ontology. The XLE parser provides the subcategorization frames of the sentences. A subcategorization frame is a type of sentence according to the number and type of syntactic arguments that co-occur with the verb. For example, *Transitive noun phrase* frame corresponds to sentences that have a subject and an object. These two arguments are noun phrases. Other examples of subcategorization frames are *Intransitive verb frame*, *Ditransitive noun phrase frame*, *Middle voice frame*, etc. On the other hand, Cyc ontology provides relations named semantic translations. These relations require subcategorization frame information to translate the output of the

parser into Cyc relations reflecting the meaning of the sentence. The translation is done using a small number of manually constructed rewrite rules.

XLE parser parses the text with Lexical Functional Grammars (LFGs) (Falk, 2011). The parser produces two mutually constraining types of output, c-structures and f-structure. C-structures, also structures of syntactic constituents are constituency trees. The f-structure analysis, on the other hand, treats the sentence as being composed of attributes, which include features such as number and tense, or functional units such as subject, predicate, or object. An example of a f-structure is presented in Figure 4.1. The parser is rule-based. Multiple parses are possible for one sentence. Our approach considers only the f-structure of the top scoring parse.

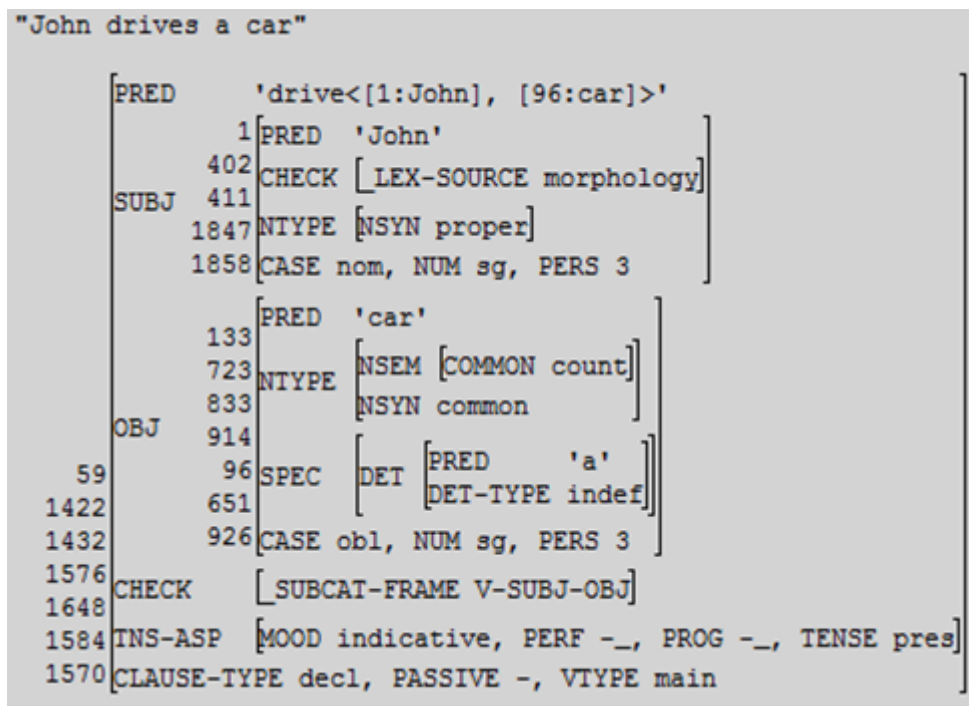


Figure 4.1: F-structure of the sentence “John drives a car.”, which was generated by the XLE parser.

4.2.1 Ontology resources

Our approach utilizes the knowledge of the Cyc ontology in two ways. First, we utilize word denotations. From the Cyc ontology, it is possible to retrieve concepts that denote a word and its part-of-speech tag pair. For example, for the pair (*play*, *verb*) the following concepts have been retrieved

- Playing
- PlayingAMusicalInstrument
- PlayingAGame
- PlayingAnAudioRecordedObject

Semantic verb translations are the second resource from Cyc that is utilized. These constructs are expressed as *verbSemTrans* relation with the following definition from Cyc:

“(verbSemTrans VERB NUM FRAME TRANS) means the translation of word sense number NUM of VERB, appearing with subcategorization frame FRAME, is TRANS.”

This is an example of *verbSemTrans* relation:

```
(verbSemTrans Drive-TheWord 0 TransitiveNPFrame
  (and
    (isa :ACTION TransportInvolvingADriver)
    (vehicle :ACTION :OBJECT)
    (driverActor :ACTION :SUBJECT)))
```

Listing 4.1: Existing verbSemTrans relation.

Last argument of the relation, the translation rule *TRANS*, is a pattern that includes free variables like *?ACTION*, *?SUBJECT*, *?OBJECT*, *?INDIRECT-OBJECT*. These variables are replaced with appropriate concepts in the translation phase.

Predicates *vehicle* and *driverActor* introduce inference in the process. Predicate *vehicle* requires *?OBJECT* to be a type of *TransportationDevice-Vehicle*. Analogously, predicate *driverActor* requires *?SUBJECT* to be a type of *Person*. Two things can happen when the final relation is being added to the ontology. If one argument cannot fulfill the semantic requirements, e.g. *?SUBJECT* is not a *Person*, then the sentence cannot be added to the ontology. Otherwise, if it cannot be proven that one of the arguments meets the requirements, e.g. it cannot be proven that *?SUBJECT* is a *Person*, then the relation is added to the ontology. Additionally, the concept replacing *?SUBJECT* becomes a type of *Person*.

Given the f-structure from Figure 4.1 and relation from Listing 4.1 the following relation is constructed:

```
(thereExists ?ACTION
  (thereExists ?OBJECT
    (and
      (isa ?ACTION TransportInvolvingADriver)
      (isa ?OBJECT Automobile)
      (vehicle ?ACTION ?OBJECT)
      (driverActor ?ACTION John))))
```

Listing 4.2: Relation for “John drives a car.”

4.2.2 Rewrite rules

In the translation phase, the system takes the f-structure as an input, and recursively executes the ordered rewrite rules. If a rewrite rule matches the f-structure on the input, it replaces it with a predicate logic pattern in the same fashion as in Section 4.1, where logic patterns replace text patterns. The first rule that satisfies the conditions is executed and no other rules are tested or executed. One rule can have multiple logic patterns, therefore the final output may consist of multiple relations, which represent all possible combinations of patterns. This is the list of rewrite rules:

1. **Rule for existing verbSemTrans relations.** Given the verb and subcategorization frame in the f-structure, the rule searches for the corresponding *verbSemTrans* relation. If the relation is found, then the f-structure is replaced by the last argument of the *verbSemTrans* relation – logic pattern. We implemented the following subcategorization frames:

- *Transitive noun phrase frame* (subject, verb, object), e.g., “John drives a car”

- *Intransitive verb frame* (subject, verb), e.g., “John swims”
- *Ditransitive prepositional phrase frame* (subject, verb, object, oblique object, preposition), e.g., “John passed the ball to Tom”
- *Copula frame* (subject, complement), e.g., “John is sad.”

2. **Rule for automatically generating verbSemTrans relations.** If the *verbSemTrans* relation was not found given the subcategorization frame and verb, then it is automatically generated. For instance the *verbSemTrans* relation in Listing 4.3 is automatically generated given the verb drive and transitive noun phrase frame. Given the verb and the frame concepts *Drive-TheWord*, *TransitiveNPFrame*, *TransportInvolvingADriver* are retrieved from the ontology using querying. The only difference from the relation in Listing 4.1 are the predicates connecting the action with the subject and object. In the generated relation, predicate *actor* is always used. This predicate is semantically poorer than predicates in existing *verbSemTrans* relations. It only means that object and subject are somehow involved in the action.

```
(verbSemTrans Drive-TheWord 0 TransitiveNPFrame
  (and
    (isa :ACTION TransportInvolvingADriver)
    (actor :ACTION :OBJECT)
    (actor :ACTION :SUBJECT)))
```

Listing 4.3: Generated verbSemTrans relation

3. **Oblique-object rule.** This rule is executed only if ditransitive prepositional phrase frame was applied before. It is used to correctly extract oblique objects from the prepositional phrase.
4. **Free variable rule.** This rule is executed when interrogative pronouns, e.g., who, what, are encountered in the interrogative sentences. These types of sentences are recognized by *CLAUSE-TYPE-INT* attribute-value pair in the f-structure. Interrogative pronouns are translated into variables. Consequently, the final logic relation is a query. For instance, for sentence “What does John drive?” the query on Listing 4.4 is generated. Variable *?OBJECT* is the only free variable in the query.

```
(thereExists ?ACTION
  (and
    (isa ?ACTION TransportInvolvingADriver)
    (vehicle ?ACTION ?OBJECT)
    (driverActor ?ACTION John))))
```

Listing 4.4: Query constructed for sentence “What does John drive?”

5. **Denotation rule.** This rule searches for word denotations (see Section 4.2.1), given the word and part-of-speech tag from f-structure. The retrieved concepts replace the input f-structure.
6. **Annotation rule.** If a particular word or phrase has not been covered by the denotation rule, information from other layers, such as named-entity layer, can be added to form a new concept. For example, if “Tiger Woods” has *Person* annotation in the named-entity layer, then the following relation is constructed

```
(thereExists ?OBJECT
  (and
    (isa ?OBJECT Person)
    (namestring ?OBJECT "Tiger Woods")))
```

7. **Noun phrase rule.** This rule splits a noun phrase into the main noun and modifiers. Each modifier is connected to the main noun with the predicate *featureOf*. For example, “white rabbit” is translated into

```
(thereExists ?OBJECT
  (and
    (isa ?OBJECT Rabbit)
    (featureOf ?OBJECT White)))
```

8. **String instance rule.** The output of this rule is a new concept created with a very semantically poor function *InstanceFn*. For example, given the word “quasicrystal” the following concept is created

```
(InstanceFn "quasicrystal")
```

This rule is always successfully applied, however the reasoning value of the concept is lower due to the nonexistent connection with the rest of ontology.

4.3 Grammar Induction

We propose a semi-automatic bootstrapping method for induction of grammar presented in Section 3.2.1. The method searches for the most frequent patterns and constructs new production rules from them. One of the main goals was to create an induction process that minimizes human involvement and maximizes the quality of semantic trees.

The input to the process, which is illustrated in Figure 4.2, is a set of predefined seed grammar rules (see Section 4.3.4) and a sample of sentences in the layered representation (see Section 3.1.1). The output of the process is a larger set of rules forming the induced grammar. One rule is added to the grammar on each iteration. At the beginning of each iteration, all the sentences are parsed with a top-down parser. The output of parsing a single sentence is a semantic tree. We distinguish two possible outcomes of the parsing: 1) the sentence was completely parsed, which is the final goal and 2) there is at least one part of the sentence that cannot be parsed. From the perspective of a parser, the second scenario happens when there is a node that cannot be parsed by any of the rules. We name these nodes – *null nodes* – and they serve as the input for the next step, the rule induction. In this step, several rules are constructed by generalization of null nodes. The generalization (see Section 4.3.2) is based on utilization of semantic annotations and bottom-up composition of the existing rules. From the induced rules, a rule with the highest frequency (the one that was generalized from the highest number of null nodes) is added to the grammar. To improve the quality of grammar, the rules are marked by the so-called property, which instructs the parser on how to use the rule (eg., use it for parsing but not for induction). The property vitally affects the result of the parsing in the following iterations potentially causing a semantic drift for the rest of the process. Consequently, a human user is needed to mark the property of each rule. The iterative process runs until a predefined stopping criteria is met. The criteria is either connected to the quality of the grammar or time limitation.

For the sake of transparency of the experiments, the human is involved in the beginning, when the seed rules are created and later when the rule properties needed to be specified. However, in another setting the user could also define new rules in the middle of the bootstrapping procedure.

In the following sections, we describe each component of the process in more detail.

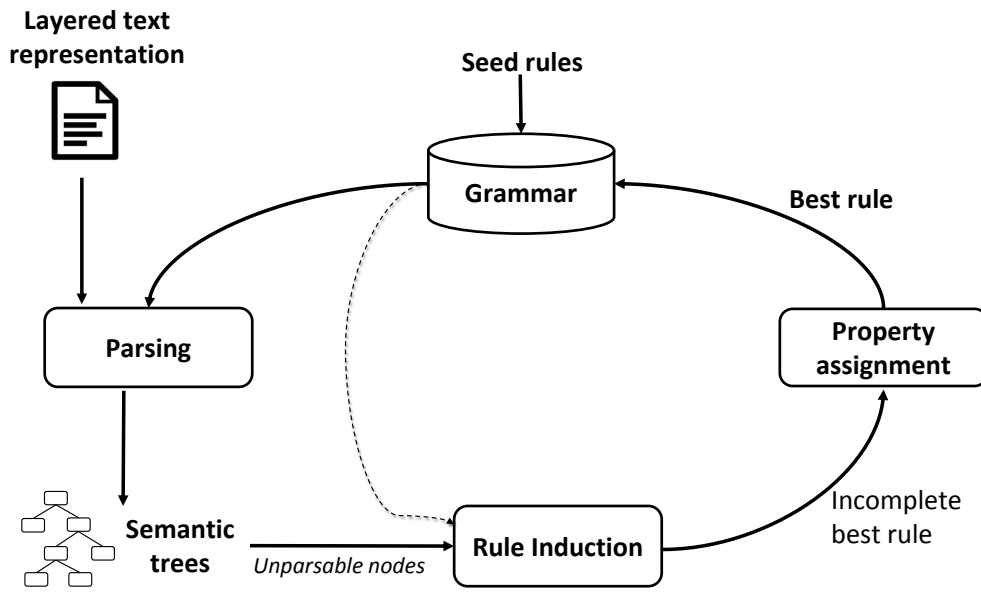


Figure 4.2: Grammar induction process.

4.3.1 Parser

A recursive descent parser with backtracking was developed for parsing. This is a top-down parser, which first looks at the higher level sentence structure and then proceeds down the parse tree to identify low level details of the sentence. The advantage of top-down parsing is the ability to partially parse sentences and to detect unparsable parts of sentences.

The parser takes a layered sentence as an input and returns a semantic tree as an output. The recursive structure of the program closely follows the structure of the parse tree. The recursive function *Parse* (see Algorithm 4.1) takes a term and a non-terminal as input and returns a parse node as an output. In order for the rule to parse the node, the left-hand side must match the input non-terminal and the right-hand side must match the layered input. In the pattern matching function *Match* (line 6), the right hand side of a rule is treated like a regular expression; non-terminals present the wildcard characters (+), which match at least one word. The terminals are treated as literal characters, which are matched against the layer that defines them. The result of a successfully matched pattern is a list of terms, where each term represents a non-terminal of the pattern. Due to ambiguity of pattern matching there might be several matches. For each of the term – non-terminal pair in every list the *parse* function is recursively called (line 10).

If none of the rules match the term, a *null node* is created and added to the list of nodes, which will be later used for grammar induction (line 18). A node is *fully parsed* if the node itself and all of its descendants are parsed. If a node is parsed and if at least one of its descendants is not parsed, then the node is *partially parsed*. All nodes that are not fully parsed are added to the list for induction.

4.3.1.1 Ambiguity

Since the grammar is ambiguous, a term can be parsed in multiple ways. There are two types of ambiguity. Two or more rules can parse the same term and one rule can parse the term in more than one way. For each ambiguity one node is created, and the best node according to *reliability measure* is selected to be the result (line 16). The measure awards

Algorithm 4.1: Pseudocode of the main function *parse* of the top-down parser.

```

1 Parse(Phrase p, Non-terminal n) output: parse node
2 nodes  $\leftarrow \{\}$ ;
3 foreach rule r of grammar do
4   if n = left side of r then
5     pattern  $\leftarrow$  right hand side of r;
6     ambiguous lists  $\leftarrow$  Match(pattern, p) ;
7     foreach term list of ambiguous lists do
8       child nodes  $\leftarrow \{\}$ ;
9       for i  $\leftarrow$  0 to size of term list do
10        child node  $\leftarrow$  Parse(term listi, pattern.non terminalsi) ;
11        add child node to child nodes;
12      add Node(type, p , r, child nodes) to nodes;
13 if nodes is empty then
14   final node  $\leftarrow$  Node(type, p , null,  $\{\}$ ) ;
15 else
16   final node  $\leftarrow$  arg maxn  $\in$  nodes r(n) ;
17 if final node is not fully parsed then
18   add final node to induction nodes ;
19 return final node

```

rules that are more specific and rules that parse more text. It is defined by

$$r(n) = \begin{cases} 1, & \text{if node is fully parsed} \\ \beta \cdot (1 - P_t(n)) + (1 - \beta) \frac{\sum_{c \in C(n)} |c| \cdot r(c)}{\sum_{c \in C(n)} |c|}, & \text{if node is partially parsed} \\ 0, & \text{if node is null} \end{cases} \quad (4.1)$$

where $P_t(n)$ is the trigger probability of the rule that parsed the node n , β is a predefined weight, $C(n)$ is the set of children of n , and $|c|$ is the length of the term of node c . The trigger probability of the rule is the probability that the right-hand side of the rule pattern matches a random term in the dataset and it is estimated after the rule is induced. If the node is partially parsed, then the measure is *strictly* between 0 and 1. The measure was defined in such a way that the more text the node parses, the higher is the reliability (the second summand in the middle row of Eq. 4.1). On the other hand, nodes with rules that are more frequently matched have a lower reliability; this penalizes rules that are very loosely defined (the first summand in the middle row of Eq. 4.1). The β parameter was set to 0.05, using grid search, with average F1 score from relation extraction experiment from Section 5.3.4 as a performance measure.

4.3.1.2 Optimizations

Since the ambiguity of the grammar may make parsing computationally intensive, several optimization techniques are used. Memoization (Norvig, 1991) is used to reduce the complexity from exponential time to $\mathcal{O}(n^3)$, where n is the length of the sentence. This was proved by Frost and Hafiz (2006). The parser does not support ϵ productions because the

grammar induction will not produce them. The patterns that do not contain terminals are the most ambiguous. At most two non-terminals are allowed, and the maximal length of the term that corresponds to the first non-terminal is three tokens. We argue that this should not be a huge limitation, since the way human languages are structured, usually two longer terms are connected with a word, like comma or a verb. These connectors should not be generalized, but rather become a terminal in the rule.

4.3.2 Rule induction

The goal of the rule induction step is to convert the null nodes from the parsing step into rules. Out of these rules, the *most frequent* one is promoted. The term from the null node is *generalized* to form the right side of the rule. The class non-terminal of the null node will present the left side of the rule. The induced rule will parse all the nodes, from which it was induced, in the following iterations. Additionally, other rules may parse the children of those nodes. Generalization is done in two steps. First, layered generalization (see Section 4.1.1.1) is applied, followed by bottom-up parsing.

4.3.2.1 Bottom-up parsing

In the bottom-up parsing step of the generalization, tokens are further generalized using a greedy bottom-up parser using the rules from the grammar. The right-hand sides of all the rules are matched against the input token term. If there is a match, the matched sub-term is *replaced* with the left side of the rule. Actually, in each iteration only all the disjunct matches are replaced. To get only the disjunct matches, overlapping matches are discarded greedily, where longer matches have the priority. This process is repeated until no more rules match the term. An example is presented in the lower part of Table 4.1.

The bottom-up parsing algorithm needs to be efficient, because the number of unparsed nodes can be large due to ambiguities in the top-down parsing. Consequently, the algorithm is greedy instead of exhaustive, and yields only one result. Aho-Corasick string matching algorithm (Aho & Corasick, 1975) is selected for matching for its ability to match all the rules simultaneously. Like the top-down parser, this parser generates partial parses because the bottom-up parser will never fully parse – the output is the same as the non-terminal type in the input node. This would generate a cyclical rule, i.e. $\langle Class \rangle ::= \langle Class \rangle$. However, this never happens because the top-down parser would have already parsed the input node.

4.3.3 Property assignment

The last step of the iteration is assigning the property to the newly induced rule. The property controls the role of the rule in the parsing and induction. The default property is *positive*, which defines the default behavior of the rule in all procedures. Rules with *neutral* property are not used in any procedure. They also cannot be re-induced. Some rules are good for parsing, but may introduce errors in the induction. These rules should be given the *non-inducible* property. For instance, rule $\langle Date \rangle ::= \langle Number \rangle$ is a candidate for the non-inducible property, since years are represented by a single number. On the contrary, not every number is a date. We attempted to introduce rules with *negative* property. Whenever such rule fully parses a node, that indicates that the current parsing path is incorrect. This allows the parser to backtrack sooner and also prevents adding null sister nodes (null sister nodes are in this case usually wrong) to the rule induction. However, it turned out that negative rules actually slow down the parsing, since the grammar gets bigger. It is better to mark these rules as neutral, so they are not used for parsing.

In our experiments, the assignment was done manually. The human user sees the induced rule and a few examples of the null nodes, from which it was induced. This should provide enough information for the user to decide in a few seconds which property to assign. After the stopping criteria is met, the iterative procedure can continue automatically by assigning a positive property to each rule. *However*, initial experimenting has shown that just a single mistake in the assignment can cause a huge drift, making all further rules wrong.

4.3.4 Seed rules

At the beginning of grammar induction, a list of seed rules is constructed. Since this step is done manually, it is reasonable to have a list of seed rules short and efficient. Seed rules can be divided in three groups: domain independent linguistic rules, class rules, top-level domain rules. *Domain independent linguistic rules*, such as

$$\begin{aligned} \langle_ \rangle &::= \text{a } \langle_ \rangle & \langle\text{Relation} \rangle &::= \langle\text{Relation} \rangle . \\ \langle_ \rangle &::= \text{an } \langle_ \rangle & \langle_ \rangle &::= \langle_ \rangle \text{ and } \langle_ \rangle \\ \langle_ \rangle &::= \text{the } \langle_ \rangle & \langle_ \rangle &::= \langle_ \rangle , \langle_ \rangle \text{ and } \langle_ \rangle , \end{aligned}$$

parse the top and mid-level nodes. They can be applied on many different datasets. *Class rules* connect class tokens, like named-entity tokens with non-terminals. For example,

$$\begin{aligned} \langle\text{Location} \rangle &::= [\text{location}]\{\text{named-entity}\} & \langle\text{Date} \rangle &::= [\text{date}]\{\text{named-entity}\} \\ \langle\text{Location} \rangle &::= [\text{Location}]\{\text{class}\} & \langle\text{Film} \rangle &::= [\text{Film}]\{\text{class}\} \end{aligned}$$

They parse the leaf nodes of the trees. On the other hand, *top-level domain rules*, define the basic structure of the sentence. For example,

$$\langle\text{Relation} \rangle ::= \langle\text{Person} \rangle \text{ is } \langle\text{Life Role} \rangle$$

As the name suggests, they parse nodes close to the root. Altogether, these rule groups parse on all levels of the tree, and may already be enough to parse the most basic sentences, but more importantly, they provide the basis for learning to parse more complex sentences.

The decision on which and how many seed rules should be defined relies on human judgment whether the current set of seed rules is powerful enough to ignite the bootstrapping procedure. This judgment may be supported by running one iteration and inspecting the top induced rules.

4.4 Taxonomy Construction from Grammar

We propose a procedure for constructing the taxonomic part of the ontology from the induced grammar. In particular, classes, instances and taxonomic relations are extracted.

First, we distinguish between instances and classes in the grammar. Classes are represented by all non-terminals and terminals that come from a layer populated with classes, for example, *named-entity* layer and *class* layer from Table 3.1. Instances might already exist in the *instance* layer, or they are created from rules, whose right hand side contains only tokens from the *lexical* layer. These tokens represent the label of the new instance. For instance rule $\langle\text{Profession} \rangle ::= \text{software engineer}$ is a candidate for instance extraction.

Furthermore, we distinguish between class and instance rules. *Class rules* have a single symbol representing a class on the right-hand side. Class rules map to *subClassOf* relations in the ontology. If the rule is positive, then the class on the right side is the subclass of the class on the left side. For instance, rule $\langle\text{Organization} \rangle ::= \langle\text{Company} \rangle$ yields relation (*subClassOf Company Organization*).

On the other hand, *instance rules* have one or more symbols representing an instance on the right side, and define the *isa* relation. If the rule is positive, then the instance on the right side is a member of a class on the left side. For instance, rule $\langle Profession \rangle ::= software\ engineer$ yields relation $(isa\ SoftwareEngineer\ Profession)$. If class or instance rule is neutral, then the relation can be treated as false. Note that many other relations may be inferred by combining newly induced relations and relations from the seed ontology. For instance, induced relation $(subClassOf\ New-class\ Seed-class)$ and seed relation $(isa\ Seed-class\ Seed-instance)$ are used to infer a new relation $(isa\ New-class\ Seed-instance)$.

4.5 Mapping Semantic Trees to Predicate Logic

Semantic trees are the intermediate representation between text and predicate logic. We present two approaches to construct predicate logic from semantic trees. In the first approach semantic rules are converted to predicate logic through rewrite rules, where each grammar rule corresponds to a logic rule. In the second approach, a machine learning model is trained to extract new relations.

4.5.1 Mapping with rewrite rules

In this approach, rewrite rules control the conversion from semantic trees into predicate logic. The approach is very similar to the one in Section 4.1. The left-hand side of a rewrite rule consists of a grammar (production) rule instead of textual pattern. Like before, the right-hand side consists of the logic, for example,

$$\begin{aligned} \langle Relation \rangle ::= \langle Person \rangle \text{ is } \langle LifeRole \rangle &\implies (isa\ ?ARG1\ ?ARG2) \\ \langle * \rangle ::= a\ \langle * \rangle &\implies ?ARG1 \\ \langle LifeRole \rangle ::= \langle LifeRole \rangle \text{ from } \langle Location \rangle &\implies (HomeLocFn\ ?ARG1\ ?ARG2) \end{aligned}$$

The conversion works from the top of the tree down recursively. Whenever a node is encountered, the algorithm searches for a rewrite rule that contains the rule of the node on the left-hand side. If there is no such rule, the conversion is not possible. Otherwise, the semantic pattern replaces the appropriate variable of the logical form. If the node is a root node, then the semantic pattern becomes the initial logic form. For leaf nodes, which correspond to atomic concepts (usually instances) in the logic form, the rewrite rules are not used. Instead, the concepts for leaf nodes are constructed given the term of the node. If the term has a valid interpretation in a layer containing concepts from the ontology, that concept can be used. Another solution is to create a new concept.

The following logic relation was constructed given the rewrite rules presented above and the semantic tree in Figure 3.1.

$$(isa\ PhilMadeira\ (HomeLocationFn\ Musician\ Nashville))$$

4.5.2 Relation extraction from semantic trees

We propose a method for learning relations from semantic trees. Given a dataset of positive relation examples that represent a predicate relation with a particular predicate, e.g. *birthPlace*, the goal is to discover new unseen relations.

The method is based on the assumption that a relation between entities is expressed in the shortest path between them in the semantic tree (Bunescu & Mooney, 2005). The input for training are sentences in layered representation, corresponding semantic trees, and relation examples. Given a relation from the training set, we first try to identify the sentence containing each argument (concept) of the relation. The relation can have one,

two, or even more arguments. Each argument is matched to the layer it belongs to. For example, strings are matched to the lexical layer; ontology concepts are matched to the layer containing such concepts. The result of a successfully matched argument is a sub-term of the sentence. In the next step, the semantic tree of the sentence is searched for a node that contains the sub-term.

At this point, each argument has its corresponding *argument node*. Otherwise, the relation is discarded from the learning process. Given the argument nodes, a minimum spanning tree containing all of them is extracted. If there is only one entity node, then the resulting subtree is the path between this node and the root node. The extracted sub-tree is converted into a *variable tree*. Variable trees are generalizations of semantic trees so that different semantic sub-trees can be converted to the same variable tree, for example see Figure 4.3.

The nodes of the semantic sub-tree are converted into variable nodes, by retaining the class and the rule of the node, and excluding the term. For argument nodes also their position in the relation is stored. Furthermore, for each link in the variable tree its relative position given the parent in the original tree is stored in the parent node. For instance, if the link was connecting the second child of its parent, number 2 is stored with the link. This information must be stored in order to maintain the structure of the original trees, even though some nodes are not part of the variable trees.

All variable trees extracted from training relations present positive examples in the training process. Negative examples of all other possible sub-trees that do not present any of the training relations. These are also converted into variable trees. Each unique node in the variable tree is a feature, analogously to bag of words, each training relation is represented by a bag of nodes. Now the problem is suitably structured for a classification algorithm, like logistic regression, to train one model for each predicate.

When predicting all possible sub-trees of the semantic tree are converted into variable trees, which are converted into features¹. If a sub-tree is predicted as positive by the model, then the terms in the leaf nodes represent the arguments of the relation.

4.6 Generating Natural Language Inference Hypotheses

In previous sections of this chapter, the approaches were using predicate logic as meaning representation, while in this section the distributed representation is used for generating text.

We propose a methodology that uses neural network models to automatically generate new Natural Language Inference (NLI) hypotheses. To measure the quality of the generated hypotheses, we propose a new evaluation approach based on the construction of new NLI datasets. Our main contribution from this section are several neural network architectures for generating NLI hypothesis.

Each example in the NLI dataset consists of two human-written sentences, a premise and a hypothesis, and a corresponding label (*entailment*, *contradiction*, *neutral*) that describes the relationship between them. Few examples are presented in Table 4.2.

Based on the hypothesis that a good dataset for training a NLI classifier consists of a variety of accurate, non-trivial and comprehensible examples we propose the following approach for the evaluation of generative models. The approach is depicted in Figure 4.4.

¹The number of leaf nodes of variable trees must match the number of arguments of the relation. Also, if the relation has two or more arguments, it is predicted several times, each time with the different argument position numbers in the argument nodes.

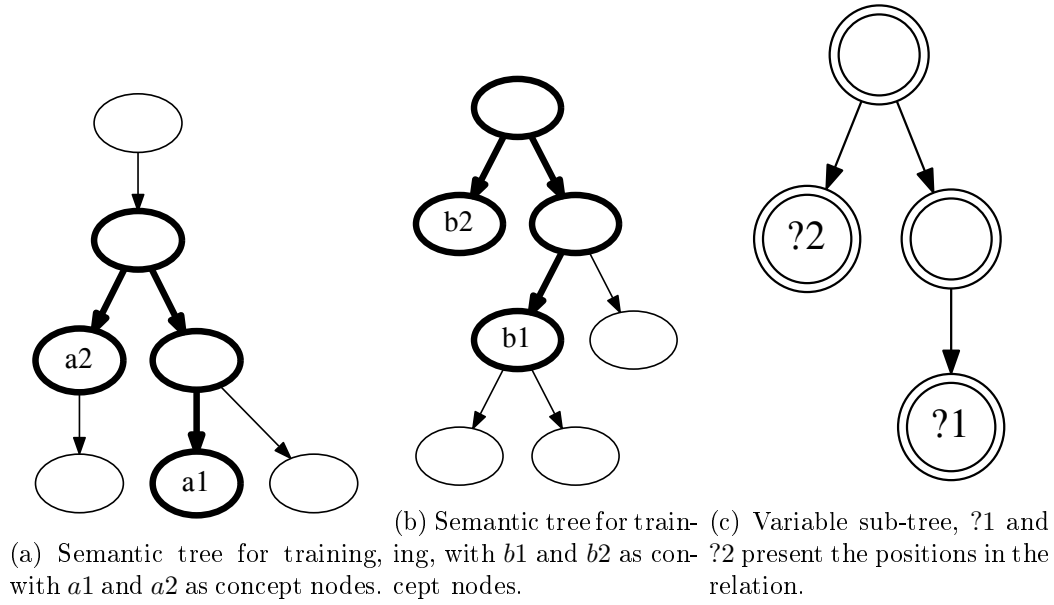


Figure 4.3: Extraction of variable trees. Given the relation $(r\ a1\ a2)$ and semantic tree on 4.3a, and relation $(r\ b1\ b2)$ and semantic tree on 4.3b, the same variable sub-tree is extracted (4.3c).

Table 4.2: Three NLI examples from SNLI corpus.

Premise	Hypothesis	Label
A person throwing a red ball in the air	The ball sails through the air	entailment
A person throwing a red ball in the air	The person throws a square	contradiction
A person throwing a red ball in the air	The ball is heavy	neutral

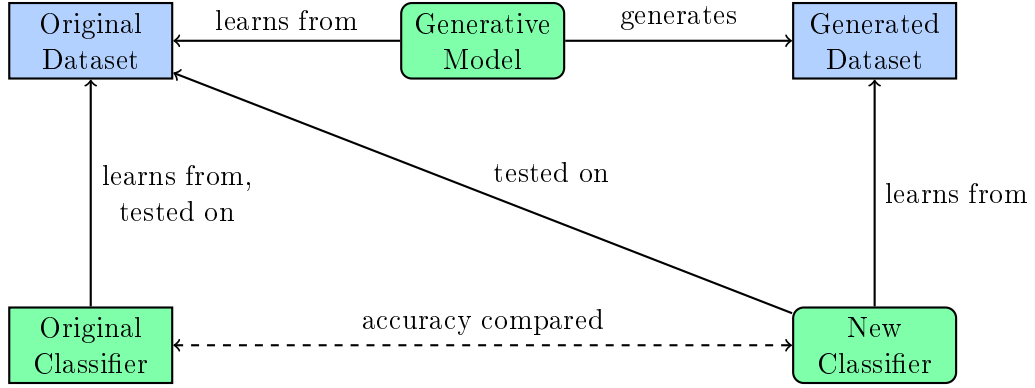


Figure 4.4: NLI dataset generation approach.

First, the generative model is trained on the original training dataset. Then, the premise and label from an example in the original dataset are taken as the input to the generative model to generate a new *random* hypothesis. The generated hypothesis is combined with the premise and the label to form a new unseen example. This is done for every example in the original dataset to construct a new dataset. Next, a classifier is trained on the new dataset. Finally, the classifier is evaluated on the original test set. The accuracy of the classifier is the proposed quality metric for the generative model. It can be compared to the accuracy of the classifier trained on the original training set and tested on the original test set.

The generative models learn solely from the original training set to regenerate the dataset. Thus, the model learns the distribution of the original dataset. Furthermore, the generated dataset is just a random sample from the estimated distribution. To determine how well did the generative model learn the distribution, we observe how *close* does the accuracy of the classifier, trained on the generated dataset, approach the accuracy of the classifier trained on the original dataset.

4.6.1 Models

We present several neural networks used for generating NLI hypotheses. The presented classification model was proposed by (Wang & Jiang, 2016), and is needed in the evaluation of the generative neural networks. We propose four generative neural network architectures for generating NLI hypotheses. Finally, we present discriminative networks, which are used for the evaluation and analysis of the hypotheses.

The premise $W^p = w_1^p w_2^p \dots w_M^p$ and hypothesis $W^h = w_1^h w_2^h \dots w_N^h$ are represented with word embeddings $X^p = x_1^p x_2^p \dots x_M^p$ and $X^h = x_1^h x_2^h \dots x_N^h$, respectively. Each x is an e -dimensional vector that represents the corresponding word, M is the length of premise, and N is the length of hypothesis. The labels (entailment, contradiction, neutral) are represented by a 3-dimensional vector Y^l if the label is the output of the model, or L if the label is the input to the model.

4.6.1.1 Classification model

The classification model predicts the label of the example given the premise and the hypothesis. The architecture of the model is presented in Figure 4.5. The embeddings of the

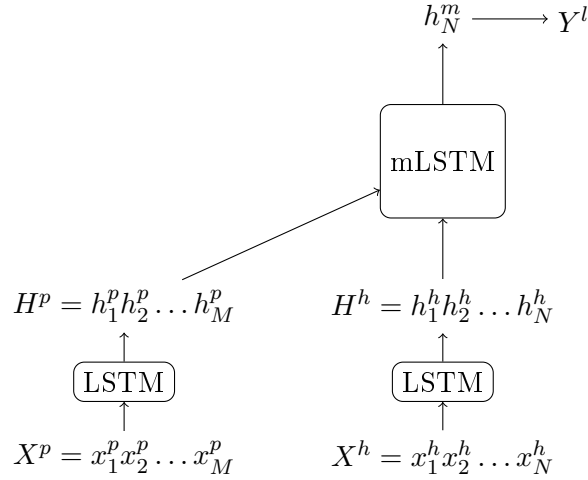


Figure 4.5: NLI classification model.

premise X^p and hypothesis X^h are the input to the first two LSTMs to obtain the hidden states of the premise H^p and hypothesis H^h .

$$H^p = LSTM(X^p) \quad H^h = LSTM(X^h) \quad (4.2)$$

All the hidden states in our models are d -dimensional unless otherwise noted. The hidden states H^p and H^h are the input to the mLSTM layer. The output of mLSTM are hidden states H^m , although only the last state h_N^m is further used. A fully connected layer transforms it into a 3-dimensional vector, on top of which the softmax function is applied to obtain the probabilities Y^l of labels.

$$Y^l = \text{softmax}(\text{Dense}_3(h_N^m)), \quad (4.3)$$

where Dense_x represents the fully connected layer, whose output size is x . Furthermore, softmax function is defined as

$$\text{softmax}(x)_j = \frac{e^{x_j}}{\sum_{k=1}^K e^{x_k}} \quad \text{for } j = 1, \dots, K \quad (4.4)$$

4.6.1.2 Generative models

The goal of the proposed generative models is to generate a diverse stream of hypotheses given the premise and the label. In this section, we present four variants of generative models, two variants of EMBEDDECODER model presented in Figure 4.6a, and two variants of ENCODERDECODER model presented in Figure 4.6b.

All models learn a latent representation Z that represents the mapping between the premise and the label on one side, and the hypothesis on the other side. The EMBEDDECODER models learn the latent representation by learning an embedding of the mapping for each training example separately. The embedding for i -th training example $Z^{(i)}$ is a z -dimensional trainable parameter vector. Consequently, $Z \in \mathbb{R}^{n \times f}$ is a parameter matrix of all embeddings, where n is the number of training examples. On the other hand, in ENCODERDECODER models latent representation is the output of the decoder.

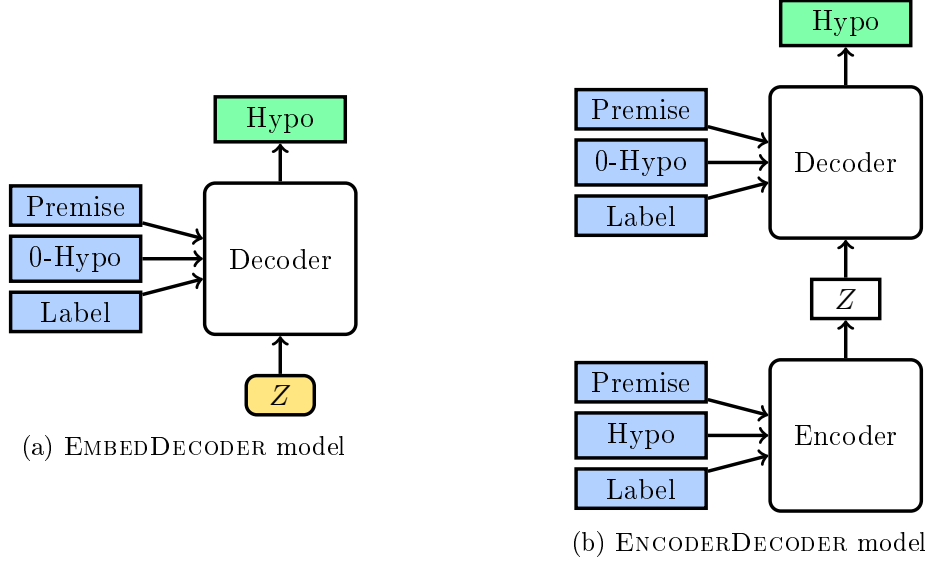


Figure 4.6: Generative model architecture. The rounded boxes represent trainable parameters, blue boxes are inputs, green boxes are outputs and the orange box represents the mapping embeddings. *0-Hypo* denotes the shifted<null>-started hypothesis. Note that in ENCODERDECODER model the latent representation Z is just a hidden layer, while in EMBEDDECODER it is a trainable parameter matrix.

The EMBEDDECODER models are trained to predict the next word of the hypothesis given the previous words of hypothesis, the premise, the label, and the latent representation of the example.

$$\theta^*, Z^* = \arg \max_{\theta, Z} \sum_{i=1}^n \sum_{k=1}^{d(W^{h^{(i)}})} \log p(w_k^{h^{(i)}} | w_{k-1}^{h^{(i)}} \dots w_1^{h^{(i)}}, W^{p^{(i)}}, L^{(i)}, Z^{(i)}, \theta) \quad (4.5)$$

where θ represents parameters other than Z , and $d(W^{h^{(i)}})$ is the length of the hypothesis $W^{h^{(i)}}$.

The ATTEMBEDDECODER, presented in Figure 4.7, is an attention-based variant of EMBEDDECODER. The same mLSTM layer is used as in the classification model. However, the initial cell state C_0 of mLSTM is constructed from the latent vector and the label input.

$$C_0 = \text{Dense}_d([Z^{(i)}, L]) \quad (4.6)$$

For the sake of simplifying the notation, we dropped the superscript (i) from the equations, except in $Z^{(i)}$, where we explicitly want to state that the embedding vector is used.

The premise and the hypothesis are first processed by LSTM and then fed into the mLSTM, like in the classification model, however here the hypothesis is shifted. The first word of the hypothesis input is an empty token <null>, symbolizing the empty input sequence when predicting the first word. The output of the mLSTM is a hidden state H^m , where each h^m represents an output word. To obtain the probabilities for all the words in the vocabulary y_k^h for the position k in the output sequence, h_k^m is first transformed into a vocabulary-sized vector, then the softmax function is applied.

$$y_k^h = \text{softmax}(\text{Dense}_V(h_k^m)), \quad (4.7)$$

where V is the size of the vocabulary. But, due to the large size of the vocabulary, a two-level hierarchical softmax (Goodman, 2001) was used instead of a regular softmax to reduce the number of parameters updated during each training step.

$$y_k^h = \text{hsoftmax}(h_k^m) \quad (4.8)$$

In the training step, the last output word y_{N+1}^h is set to $\langle \text{null} \rangle$, while in the generating step, it is ignored.

In the EMBEDDECODER model without attention, BASEEMBEDDECODER, the mLSTM is replaced by a regular LSTM. The input to this LSTM is the shifted hypothesis. But, here the premise is provided through the initial cell state C_0 . Specifically, the last hidden state of the premise is merged with class input and the latent representation, then fed to the LSTM.

$$C_0 = \text{Dense}_{d'}([Z^{(i)}, L, h_M^p]) \quad (4.9)$$

In order not to lose information d' was picked to be equal to the sum of the sizes of $Z^{(i)}$, L and h_M^p . Thus, $d' = f + 3 + d$. Since the size of C_0 is d' , the output vectors of the LSTM are also the size of d' .

We also present two variants of ENCODERDECODER models, a regular one BASEENCODERDECODER, and a regularized one VARENCODERDECODER, which is based on the Variational Bayesian approach. As presented in Figure 4.6b, all the information (premise, hypothesis, label) is available to the encoder, whose output is the latent representation Z . On the other hand, the decoder is provided with the same premise and label, but the hypothesis is shifted. This forces the encoder to learn to encode only the missing information – the mapping between premise-label pair and the hypothesis. The encoder has a similar structure as the classification model in Figure 4.5. Except that the label is connected to the initial cell state of the mLSTM

$$C_0 = \text{Dense}_d(L), \quad (4.10)$$

and the output of mLSTM h_N^m is transformed into latent representation Z

$$Z = \text{Dense}_z(h_N^m). \quad (4.11)$$

The decoder is the same as in EMBEDDECODER.

The VARENCODERDECODER model is based on Variational Autoencoder from (Kingma & Welling, 2013). Instead of using single points for latent representation as in all previous models, the latent representation in VARENCODERDECODER is presented as a continuous variable $Z \sim \mathcal{N}(Z_\mu, Z_\sigma)$. Thus, the mappings are presented as soft elliptical regions in the latent space, instead of single points, which forces the model to fill up the latent space (Bowman, Vilnis, et al., 2015). Both Z_μ and Z_σ are calculated from the output of the encoder using two different fully connected layers.

$$Z_\mu = \text{Dense}_z(h_N^m), \quad Z_\sigma = \text{Dense}_z(h_N^m).$$

To sample from the distribution the reparametrization trick is applied

$$Z = Z_\mu + Z_\sigma \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbb{I}) \quad (4.12)$$

When training, a single sample is generated per example to generate Z .

As in (Kingma & Welling, 2013), the following regularization term is added to the loss function

$$\frac{1}{2}(1 + \log(Z_\sigma^2) - Z_\mu^2 - Z_\sigma^2). \quad (4.13)$$

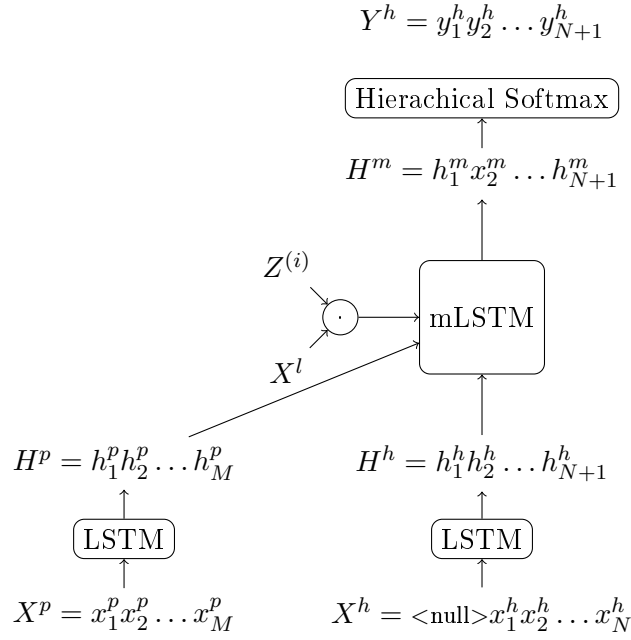


Figure 4.7: ATTEMBEDDECODER model.

4.6.1.2.1 Generating hypotheses In the generation phase only the decoder of a trained generative model is used. It generates a hypothesis given the premise, label, and a randomly selected latent vector $Z^{(*)}$. A single word is generated in each step, and it becomes the hypothesis input in the next step.

$$x_k^h = \text{embedding}(\arg \max y_k^h) \quad (4.14)$$

We also used beam search to optimize hypothesis generation. Similarly as in Sutskever et al. (2014), a small number of hypotheses are generated given a single input, then the best is selected. In k -beam search, in each time step k best partial hypotheses are expanded by all the words in the vocabulary producing kV partial hypothesis. Out of these k best partial hypotheses are selected for the next step according to the joint probability of each partial hypothesis. Thus, when k is 1, the procedure is the same as the one presented in Eq 4.14. The generation ends when $\langle \text{null} \rangle$ symbol is encountered or maximum hypothesis length is reached². The random latent vector $Z^{(*)}$ is selected randomly from a normal distribution $\mathcal{N}(0, \sigma)$, where σ is the standard deviation of Z .

4.6.1.3 Discriminative model

The discriminative model is used to measure the distinguishability between the original human written sentences and the generated ones. Higher error rate of the model means that the generative distribution is similar to the original distribution, which is one of the goals on the generative model. The model is based on Generative Adversarial Nets (Goodfellow et al., 2014), where in a single network the generative part tries to trick the discriminative part by generating images that are similar to the original images, and the

²In beam search mode the process stops when all k hypotheses reach the $\langle \text{null} \rangle$ symbol or maximum hypothesis length is reached

discriminative part tries to distinguish between the original and generated images. Due to the discreteness of words (the output of our generative model) it is difficult to connect both the discriminative and generative part in a single differentiable network, thus we construct them separately. The generative models have already been defined in Section 4.6.1.2. Here we define the discriminative model.

The discriminative model D takes sequence X and processes it with LSTM and fully connected layer

$$D(X) = \sigma(\text{Dense}_1(\text{LSTM}(X))) \quad (4.15)$$

In the training step, one original sequence X_{original} and one generated sequence $X_{\text{generated}}$ are processed by the discriminative model. The optimization function maximizes the following objective

$$\log(D(X_{\text{original}})) + \log(1 - D(X_{\text{generated}})) \quad (4.16)$$

In the testing step, the discriminative model predicts correctly if

$$D(X_{\text{original}}) > D(X_{\text{generated}}) \quad (4.17)$$

Chapter 5

Evaluation

In this chapter, we present our next contribution, the evaluation of the proposed approaches. We start by preliminary evaluation of approaches based on mapping to predicate logic in Section 5.1 and Section 5.2. We continue with a more thorough evaluation of the remaining approaches; the approach based on grammar induction in Section 5.3 and the approach for constructing NLI datasets in Section 5.4. In the last section all approaches are compared on multiple levels.

5.1 Mapping to Predicate Logic with Textual Patterns

This section presents an experiment to evaluate the approach for mapping to predicate logic through textual patterns described in Section 4.1. First, textual patterns are generated and filtered using the statistics described in Section 4.1.1.2. Then, four rules were constructed using those patterns. Finally, the rules were applied on the corpus and evaluated.

The corpus for evaluation contained about half a million English news articles, which is about 14 million sentences. The corpus was transformed into the layered representation with a lexical and named-entity layer. We obtained named-entity annotations of the following types: *person*, *location*, *organization*, *date*, and *money*. The annotations were obtained using Enrycher (Štajner et al., 2010). Next, sentences were generalized and n-grams were extracted.

The extraction produced two sets of n-grams: a set of n-grams containing six tokens or less, and a set of n-grams that occurred at least twice and contained ten tokens or less. From these n-grams, we generated all possible n-grams with one gap that had maximally five non-gap tokens.

For all the n-grams we calculated statistics presented in Section 4.1.1.2. The statistics were manually evaluated, and for each length n-grams were filtered and sorted into a table. Patterns with a frequency lower than 20 ($Fq < 20$) were discarded since rules that trigger more frequently are preferred. Patterns that had two or three arguments ($Arg = 2$ or $Arg = 3$) were considered, others were discarded. We observed that the number of stop words (SW) did not affect the quality of the patterns. The output table was sorted according to the minimal token frequency (MTF). The patterns with high (MTF) were usually too generic for a valuable rule. We noticed that n-grams with higher normalized expectation (NE) were more valuable. An example of a table with n-grams of length 6 is presented in Table 5.1.

The last part of the experiment evaluates the four rules in Table 5.2. The rules were constructed and applied on text. The Cyc ontology is used to build logic patterns and new concepts and relations were added to this ontology. The testing corpus contained 7500 news articles published the same day. We constructed four rules and applied them on the

Table 5.1: Part of the table representing 6-gram patterns and their statistics.

Pattern	Fq	Arg	SW	MTF	NE
<i>[PERSON] , executive director of [ORGANIZATION]</i>	21	2	2	11699	0.48
<i>'s hospital in [LOCATION] , [LOCATION]</i>	22	2	2	11917	0.57
<i>to [PERSON] parents , [PERSON] was</i>	40	2	3	12020	0.56
<i>death by [PERSON] parents , [PERSON]</i>	24	2	2	12020	0.28
<i>[PERSON] parents , [PERSON] and [PERSON]</i>	22	3	2	12020	0.51
<i>[PERSON] have no idea what [PERSON]</i>	20	2	3	12449	0.55
<i>([ORGANIZATION]) - [PERSON] scored</i>	22	2	3	12514	0.73
<i>victory over the [ORGANIZATION] on [DATE]</i>	55	2	3	12626	0.65
<i>, died [DATE] , at [ORGANIZATION]</i>	61	2	3	12822	0.71
<i>died [DATE] , at [ORGANIZATION] in</i>	45	2	3	12822	0.73
<i>[PERSON] was a member of [ORGANIZATION]</i>	38	2	3	13399	0.62

Table 5.2: Selected four mapping rules.

Nº	Rule
1	$\frac{[ORGANIZATION] \text{ coach } [PERSON] \implies (hasCoach \text{ ?ORGANIZATION ?PERSON})}{[PERSON] , [POSITION] \text{ of } [ORGANIZATION] \implies (positionOfPersonInOrganization \text{ ?PERSON ?ORGANIZATION ?POSITION})}$
2	$\frac{[PERSON] \text{ said } [STRING] \implies (thereExists \text{ ?INFORMING (and (isa ?INFORMING Informing) (senderOfInfo ?INFORMING ?PERSON))})}{[PERSON] \text{ father } \implies (FatherFn \text{ ?PERSON})}$
3	(infoTransferred-NLString ?INFORMING ?STRING))
4	$[PERSON] \text{ father } \implies (FatherFn \text{ ?PERSON})$

test corpus. If none of the rules were successfully applied on a particular term, then the ontology was searched for concepts having the same label as argument. If no concepts have been found, a new concept was created. Per one rule application there might have been multiple relations constructed; one for each possible combination of retrieved concepts.

One human evaluator examined all the generated relations. The evaluator was given the sentence containing the rule match and the constructed relations to subjectively decide whether the meaning of one of the relations is also found in the sentence. Not having a system for word sense disambiguation integrated in our system, we considered the match valid if one of the relations reflects the meaning. Table 5.3 presents the results for the four constructed rules. *Matches* is the number of times the rule was successfully applied to text. *ValidM* is the number of valid matches. Matches that have more than one relation are ambiguous matches (AmbigM), while *TotalR* is the total number of relations per rule. The last two columns represent how many unique concepts that appear in the arguments were already in the ontology (*ExConcepts*), while *NewConcepts* column represents the number of newly created concepts that appear in the arguments. Rule 4 does yield compound concepts instead of relations. Compound concepts cannot be added to the ontology by themselves, only if they are part of a relation. Therefore, we considered a match valid only if it is nested inside a relation.

Table 5.3: Evaluation of four selected rules.

Nº	Matches	ValidM	ValidM%	AmbigM	TotalR	ExConcepts	NewConcepts
1	24	16	0.67	0	19	29	9
2	38	20	0.53	2	30	14	70
3	379	337	0.89	37	830	227	35
4	147	1	-	-	-	24	78

Table 5.4: Evaluation of f-structure quality.

Quality class description	% of sentences
The parse is completely correct.	76
The parse is almost correct. It is good enough to be further used.	8
The parse is mainly incorrect.	16

5.2 Mapping to Predicate Logic Using Syntax Parser

In this section, the approach from Section 4.2, which utilizes the syntax parser to construct predicate logic, is evaluated. A controlled set of sentences was parsed with XLE parser and using the proposed rule-based approach meaning representations in predicate logic were constructed. The predicate logic relations were added to Cyc ontology.

A dataset of articles was obtained from IJS Newsfeed (Trampuš & Novak, 2012) and preprocessed with Enrycher to obtain two-layer (lexical and named-entity) representation. Given the topic classification from Enrycher, only business domain English news were retained. To improve the quality of parses, very long and very short sentences were excluded. Manual inspection has shown that sentences having from 7 to 15 tokens had the best chance of being correctly parsed. Therefore, only those sentences were retained. Furthermore, sentences containing non-alphabetic characters including commas were excluded. From a total of 40624 articles, 19443 articles were extracted. These sentences had 12245 named-entity layer tokens. Logic relations were constructed for 1000 randomly selected sentences. 326 sentences had at least one logic relation. Out of these, 225 sentences had multiple relations. 204 sentences had at least one relation that was added to the ontology, which means there were no contradictions.

For manual evaluation 50 sentences that had at least one logic relation were randomly selected. Only small number of relations were completely error-free. Therefore, different aspects of the process were evaluated separately. One human evaluator evaluated the quality of the XLE f-structures (see Table 5.4), qualities of phrase denotations (see Table 5.5), the quality of the meaning representation structure and the semantic value of predicates (see Table 5.6). Since there are multiple (ambiguous) relations possible per one sentence, the evaluator has chosen the most correct one. If some parts of sentences, like adverbs, did not appear in the meaning representation, the translation was not penalized. Then, the evaluator for each aspect selects a description that best describes the example given a list of predefined descriptions.

Table 5.4 shows that most of the parses are correct. Some of them are almost correct meaning that there is a minor error, which should not extensively affect the meaning representation. On the contrary, the last group represents the incorrect parses, which should not be further used.

Table 5.5 shows how the argument phrases were transformed into logic form. Argument phrases present arguments of the first two rules – the *verbSemTrans* rules. These phrases

Table 5.5: Evaluation of phrase denotation quality.

Denotation quality	Description	Phrases %
Correct	The phrase is denoted by the correct concept. Denotation rule (5) was used.	42
Named Entity	Annotation rule (6) was used.	1
Missing	String instance rule (8) was used. A corresponding concept is missing in the ontology.	18
Poor	String instance rule (8) was used. The phrase should be further split into smaller denotable phrases and then combined into a compound concept.	28
Incorrect	The phrase was incorrectly denoted by the denotation rule (5).	11

Table 5.6: Evaluation of meaning representation structure and semantic value of predicates.

Quality class description	% of sentences
The structure is good, predicates are semantically rich.	26
The structure is good, not all predicates are semantically rich.	22
There is a minor error in the structure.	22
The structure is completely wrong.	30

are usually noun phrases. The last four rows present phrases, for which the sufficient concepts did not exist in the ontology. The missing vocabulary is either replaced with semantically poorer concepts (rows 2,3,4) or the denotation is incorrect (last row).

While Table 5.5 investigates the quality of concepts that correspond to the leaf parts of the parse tree, Table 5.6 presents the results of the quality of the higher level of parse trees, which is covered by *verbSemtrans* rules (1, 2) and the noun phrase rule (7). The evaluator assessed the structure and semantical richness of those parts of the meaning representations. Predicates are not semantically rich if rule 2 or 7 is used. Minor errors in the structure are, for example, when a less important part of the sentence is not represented in the meaning representation.

5.3 Grammar Induction

We present experiments evaluating the approach based on grammar induction, which is presented in Section 4.3. We have conducted experimentation on dataset presented in Section 5.3.1. First, the grammar was induced on the Wikipedia dataset to present the grammar’s characteristics, and the scalability of the approach (see Section 5.3.2). In the next experiment, we present a method for discovering less prominent instances (see Section 5.3.3). The last experiment was the supervised learning of DBpedia relations (see Section 5.3.4).

5.3.1 Datasets

The datasets for experiments were constructed from English Wikipedia, DBpedia and Freebase ontologies. First sentences of Wikipedia pages describing people were taken as

the *textual dataset*, while DBpedia relations expressing facts about the same people were taken as the *dataset for supervised relation learning*. Note that each DBpedia instance corresponds to a Wikipedia page. A set of person instances was identified by querying DBpedia for instances that are a type of person. The Wikipedia pages representing these entities were parsed by the in-house Wikipedia markup parser¹ to convert the markup into plain text. Furthermore, the links to other Wikipedia pages were retained. Here is an example of a sentence in plain text:

“Victor Francis Hess (24 June 1883 – 17 December 1964) was an Austrian-American physicist, and Nobel laureate in physics, who discovered cosmic rays.”

Using the Stanford OpenNLP toolkit (Manning et al., 2014) we obtained sentence and token splits, and named-entity annotations. Only the first sentence of each page was retained and converted to the proposed layered representation. The layered representation contains five layers: *lexical* (plain text), *named-entity* (named entity type), *wiki-link* (Wikipedia page in the link – DBpedia instance²), *dbpedia-class* (class of Wikipedia page in Dbpedia) and *freebase-class* (class of Wikipedia page in Freebase). Freebase also contains its own classes of Wikipedia pages. For the last two layers there might be several classes per Wikipedia page. Only one was selected using a short priority list of classes. If none of the categories is on the list, then the category is chosen at random. After comparing the *dbpedia-class* and *freebase-class* layers, only *freebase-class* was utilized in the experiments because more *wiki-link* tokens have a class in *freebase-class* layer than in *dbpedia-class* layer.

As a result, almost 1.1 million sentences were extracted and annotated. The average length of a sentence is 18.3 words, while the median length is 13.8 words. There are 2.3 links per sentence.

The dataset for supervised relation learning contains all relations where a person instance appears as a subject in the DBpedia relation. For example,

(dbpedia-owl:birthDate dbpedia:Victor_Francis_Hess 1883-06-24)

In all relations, there are 119 different unique predicates, having from just a few relations to a few million relations. Since DBpedia and Freebase are available in RDF format, we used the RDF store for querying and for storage of existing and new relations.

5.3.2 Grammar induction experiments

The grammar was induced on 10,000 and tested on 100,000 other random sentences taken from the dataset. The manual work in the experiments was done by one of the authors in (Starc & Mladenić, 2017), referred to as the human user. First, a list of 45 seed nodes was constructed by the human user. There were 22 domain independent linguistic rules, 17 category rules and 6 top-level rules created, which was enough to start the bootstrapping process. The property assignment was also done by the human user. In every iteration, the best rule is shown together with the number of nodes it was induced from, and ten of those nodes together with the sentences they appear in. All this information was presented in the textual form. The goal was set to stop the iterative process after two hours. We estimated this is the right amount of time, in which the human user can consistently provide quality feedback.

¹The markup parsing of Wikipedia markup is non-trivial. To extract the plain text, many elements have been discarded, like images, tables, and some elements that appear in the middle of sentence, like pronunciation and citation.

²Each Wikipedia page has a corresponding DBpedia instance.

Table 5.7: A sample of induced rules.

Nº	rule	property
1	<PersonAttr> ::= born <Date>	none
101	<LifeRole> ::= born in <Location>	none
201	<Location> ::= <Location> from <Date>	neutral
301	<Person> ::= <Location> from <Date>	neutral
401	<OrderOfChivalry> ::= (<PersonAttr>)	neutral
501	<LifeRole> ::= who <Action> in <Event>	none
601	<Date> ::= the university	neutral

Table 5.8: Statistics of test set.

Grammar	
positive rules	231
non-inducible rules	21
neutral rules	437
Parsing	
fully parsed sentences	25.63%
avg. coverage	78.52%
avg. tree depth	6.96
avg. number of leaf nodes	6.69
avg. number of null leaf nodes	1.98
avg. number of operations	320.3
avg parsing time	0.16 ms

5.3.2.1 Results

There were 689 new rules created. A sample of them is presented in Table 5.7. Table 5.8 presents the distributions of properties and parsing statistics. Around 36% of rules were used for parsing (non-neutral rules). Together with the seed rules there are 297 rules used for parsing. Different properties are very evenly dispersed across the iterations. Using the procedure for conversion of grammar rules into taxonomy presented in Section 4.4, **33** classes and subClassOf relations and **95** instances and isa relations were generated.

The sentences in the test set were parsed with the new grammar. A few statistics are presented in Table 5.8. More than a quarter of sentences were fully parsed, meaning that they do not have any null leaf nodes. Coverage represents the fraction of words in a sentence that were parsed (words that are not in null-nodes). The number of operations shows how many times was the *Parse* function called during the parsing of a sentence. It is highly correlated with the time spent for parsing a sentence, which is on average 0.16ms. This measurement was done on a single CPU core. Consequently, it is feasible to parse a collection of a million sentences, like our full dataset. The same statistics were also calculated on the training set, the numbers are very similar to the test set. The fully parsed % and coverage are even slightly lower than on the test set.

Some of the statistics were calculated after each iteration, but only when a non-neutral rule was created. The graphs in Figure 5.1 show how the statistics have changed over the course of the grammar induction. Figure 5.1a shows that the coverage and the fraction of fully parsed sentences are correlated and they grow very rapidly at the beginning. Then the growth starts to slow down, which indicates that there is a long tail of unparsed

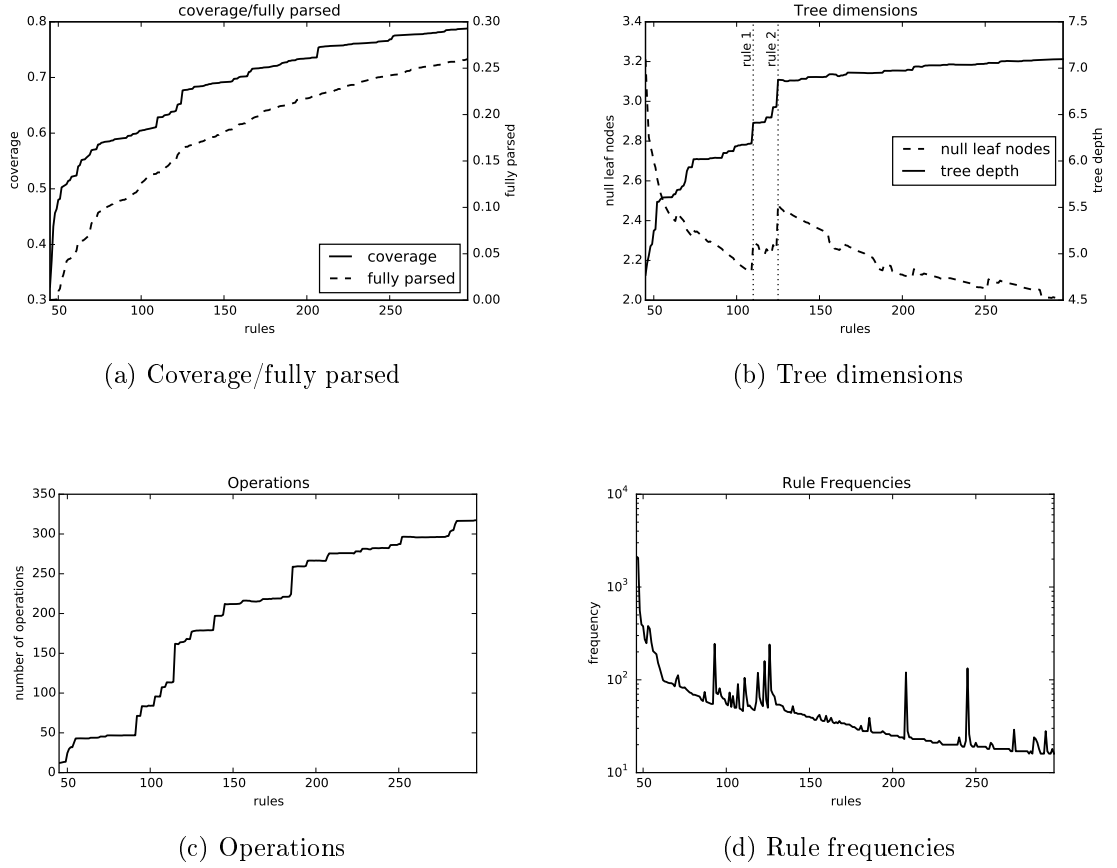


Figure 5.1: Statistics of the iterative process.

nodes/sentences. In the following section, we present an instance learning method, which deals with the long tail. Furthermore, the number of operations per sentence also slows down (see Figure 5.1c) with the number of rules, which gives a positive sign of retaining computational feasibility with the growth of the grammar. Figure 5.1b somewhat elaborates the dynamics of the grammar induction. In the earlier phase of induction many rules that define the upper structure of the tree are induced. These rules can rapidly increase the depth and number of null nodes, like *rule 1* and³ *rule 2*. They also explain the spikes on Figure 5.1d. Their addition to the grammar causes some rules to emerge on the top of the list with a significantly higher frequency. After these rules are induced the frequency gets back to the previous values and slowly decreases over the long run.

5.3.3 Instance extraction

We present an experiment with a method for discovering new instances, which appear in the long tail of null nodes. Note that the majority of the instances were already placed in the ontology by the method for extracting the taxonomy. Here, less prominent instances are extracted to increase the coverage of semantic parsing. The term and the class of the null node will form an *isa* relation. The class of the node represents the class of

³rule1 - $\langle LifeRole \rangle ::= \langle LifeRole \rangle$ in $\langle Location \rangle$,
rule2 - $\langle LifeRole \rangle ::= \langle LifeRole \rangle$ of $\langle Organization \rangle$

Table 5.9: Instance relations per category.

Class	Relations	Accuracy %
Life role	15356	60
Person	5427	49
Order of chivalry	1319	32
Date	1310	46
Action	967	78
Field of study	25	80

the relation. The terms are converted to instances. They are first generalized on the layer level (see Section 4.1.1.1). The goal is to exclude non-atomic terms, which do not represent instances. Therefore, only terms consisting of one *wiki-link* token or exclusively of lexical tokens are retained. The relations were sorted according to their frequency. We observe that the accuracy of relations drops with the frequency. Therefore, relations that occurred less than three times were excluded. The number and accuracy for six classes is reported in Table 5.9. Other classes were less accurate. For each class, the accuracy was manually evaluated on a random sample of 100 instance relations. Taking into account the estimated accuracy, there were more than 13.000 correct *isa* relations.

5.3.4 Relation extraction

We present an experiment of the relation extraction methods presented in Section 4.5.2. The input for the supervision is the DBpedia relation dataset from Section 5.3.1. The first argument of every relation is a person DBpedia instance, which corresponds to a particular person Wikipedia page. In the beginning, the first sentence of that Wikipedia page is identified in the textual dataset. If the last argument of this relation matches a sub-term of this sentence, then the relation is eligible for experiments. We distinguish three types of values in objects. DBpedia resources are matched with a *wiki-link* layer. Dates get converted to the format that is used in English Wikipedia. They are matched against the *lexical* layer. Strings are also matched to the *lexical* layer.

Only predicates that have 200 or more eligible relations have been retained. This is 74 out of 119 predicates. These predicates have 23.8% of eligible relations, meaning that roughly a quarter of these DBpedia person relations are expressed in the first sentence of their Wikipedia page. The macro average number of eligible relations is 17.7%. In this case, the percentage is calculated for each predicate, then the average is taken.

The prediction problem is designed in the following way. Given the predicate and the first argument of the relation (person), the model predicts the second argument of the relation. Because not all relations are functional, like for instance *child* relation, there can be several values per predicate–person pair; on average there are 1.1. Since only one argument of the relation is predicted, the variable trees presented in Section 4.5.2, are paths from the root to a single node. Analysis of variable tree extraction shows that on average 60.8% of eligible relations were successfully converted to variable trees (the second argument term exactly matches the term in the node). Others were not converted because 8.2% of the terms were split between nodes and 30.9% terms are sub-terms in nodes instead of complete terms. Measuring the diversity of variable trees shows that a distinct variable tree appeared 2.7 times on average.

Several models based on variable trees were trained for solving this classification problem:

- *Basic (Basic model)* – The model contains only positive training variable trees. In the prediction, if the test variable tree matches one of the trees in the model, then the example is predicted positive.
- *Net (Automaton model)* – All positive variable trees are paths with start and end points. In this model they are merged into a net, which acts as a deterministic automaton. If the automaton accepts the test variable tree, then it is predicted positive. An example of automaton model is presented in Figure 5.2.
- *LR (Logistic regression)* – A logistic regression model is trained with positive and negative examples, where nodes in variable trees represent features.
- *LRC (Logistic regression + Context nodes)* – All leaf nodes that are siblings of any of the nodes in the variable tree are added as features to the *LR* model.
- *LRCL (Logistic regression + Context nodes + Lexical Tokens)* – Tokens from the lexical layer of the entity nodes are added to the *LRC* as features.

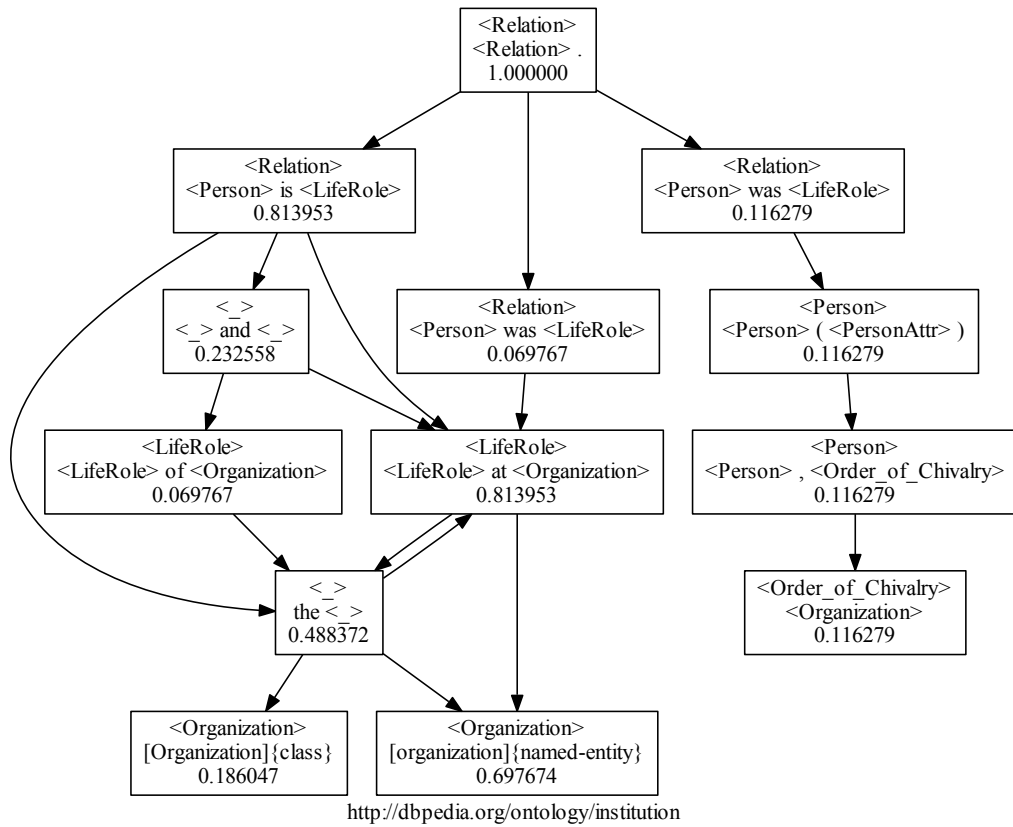


Figure 5.2: The automaton model for relation extraction. This example presents the automaton model for *institution* relation. On the bottom of each node, the fraction of training variable trees that contain this node is displayed.

Table 5.10: Performance of various relation extraction models.

Model	Precision	Converted Recall	Recall	Converted F1	F1
Basic	72.2	48.3	28.2	52.1	36.4
Net	61.6	77.6	42.5	63.3	46.9
LR	71.9	84.2	50.2	77.0	55.5
LRC	72.4	84.4	50.3	77.3	55.7
LRCL	76.2	85.5	50.7	80.0	57.1

For training all or a maximum of 10,000 eligible relations were taken for each of 74 relation types. A 10-fold cross validation was performed for evaluation. The results are presented in Table 5.10. The converted recall and converted F1 score presents recall and F1 on converted examples, which are the one, where relations were successfully converted into variable trees. The performance increases with each model, however the interpretability decreases. We also compared our method to the conditional random fields (CRF), a method known to perform well in sequence tagging. In the CRF method, tokens from all layers with a window size 7 were taken as features for sequence prediction. On the converted examples CRF achieved F1 score of 80.8, which is comparable to our best model’s (LRCL) F1 score of 80.0.

5.4 Dataset Generation

We present the evaluation of the approach for generating NLI hypotheses presented in Section 4.6.

5.4.1 Experiment details

All the experiments are performed on the SNLI dataset. There are 549,367 examples in the dataset, divided into training, development and test set. Both the development and test set contain around 10,000 examples. Some examples are labeled with ‘-’, which means there was not enough consensus on them. These examples are excluded. Also, to speed up the computation we excluded examples, which have the premise longer than 25 words, or the hypothesis longer than 15 words. There were still 92.5% remaining examples. Both premises and hypothesis were padded with <null> symbols (empty words), so that all premises consisted of 25 words, and all hypotheses consisted of 15 tokens.

We use 50-dimensional word vectors⁴ trained with GloVe (Pennington, Socher, & Manning, 2014). For words without pretrained embeddings, the embeddings are randomly selected from the normal distribution. Word embeddings are not updated during training.

For optimization, the Adam method (Kingma & Ba, 2014) was used with the suggested hyperparameters⁵.

Classification models are trained until the loss on the validation set does not improve for three epochs. The model with the best validation loss is retained.

Generative models are trained for 20 epochs, since it turned out that none of the stopping criteria was useful. With each generative model a new dataset is created. The new dataset consists of a training set, which is generated using examples from the original training set, and a development set, which is generated from the original development set.

⁴<http://nlp.stanford.edu/data/glove.6B.zip>

⁵As suggested in (Kingma & Ba, 2014) β_1 is set to 0.9 and β_2 is set to 0.999.

The beam size for beam search was set to 1. The details of the decision are presented in Section 5.4.2.

Some datasets were constructed by filtering the generated datasets according to various thresholds. Thus, the generated datasets were constructed to contain enough examples, so that the filtered datasets had at least the number of examples as the original dataset. In the end, all the datasets were trimmed down to the size of the original dataset by selecting the samples sequentially from the beginning until the dataset had the correct size. Also, the datasets were filtered so that each of the labels was represented equally. All the models including classification and discriminative models were trained with hidden dimension d set to 150, unless otherwise noted.

Our implementation is accessible at http://github.com/jstarc/nli_generation. It is based on libraries *Keras*⁶ and *Theano* (Theano Development Team, 2016).

5.4.2 Preliminary evaluation

First, the classification model ORIGCLASS was trained on the original dataset. This model was then used throughout the experiments for filtering the datasets, comparison, etc. Notice that we have assumed ORIGCLASS to be ground truth for the purpose of our experiments. However, the accuracy of this model on the original test set was 81.3%, which is less than 86.1%, which was attained by *mLSTM* ($d=150$) model in (Wang & Jiang, 2016). Both models are very similar, including the experimental settings, however ours was trained and evaluated on a slightly smaller dataset.

Several instances of the proposed ATTEMBEDDECODER models with various latent dimensions $z \in [2, 4, 8, 16, 32, 147]$ were first trained and then used to generate new datasets. A couple of generated examples are presented in Table 5.11.

Figure 5.3 shows the accuracies of the generated development datasets evaluated by the ORIGCLASS. The maximum accuracy of 64.2% was achieved by EMBEDDECODER ($z=2$), and the accuracy is decreasing with the number of dimensions in the latent variable. The analysis for each label shows that the accuracy of contradiction and neutral labels is quite stable, while the accuracy of the entailment examples drops significantly with latent dimensionality. One reason for this is that the hypothesis space of the entailment label is smaller than the spaces of other two labels. Thus, when the dimensionality is higher, more creative examples are generated, and these examples less often comply with the entailment label.

Since none of the generated datasets' accuracies is as high as the accuracy of the ORIGCLASS, we used it to filter the datasets subject to the various prediction thresholds. The examples from the generated dataset were evaluated by ORIGCLASS and if the probability of the label of the example exceeded the threshold $t \in [0.0, 0.3, 0.6, 0.9]$, then the example was retained. For each filtered dataset a classifier was trained. Figure 5.4a shows the accuracies of these classifiers on the original test set. Filtering out the examples that have incorrect labels (according to the ORIGCLASS) improves the accuracy of the classifier. However, if the threshold is set too high, the accuracy drops, since the dataset contains examples that are too trivial. Figure 5.4b, which represents the accuracy of classifiers on their corresponding generated development sets, further shows the trade-off between the accuracy and triviality of the examples. The classifiers trained on datasets with low latent dimension or high filtering threshold have higher accuracies. Notice that the training dataset and test dataset were generated by the same generative model.

⁶<http://keras.io>

⁷Latent dimension $z = 147$ is the largest dimension so that there is no reduction in dimensionality in Equation 4.6, therefore $z + c = d$, where c is the number of labels.

Table 5.11: Generated examples to illustrate the proposed approach.

z	Premise	A person throwing a yellow ball in the air.
2	neutral	Someone is playing basketball.
	contradiction	A person is sleeping in a chair.
	entailment	A person is throwing a ball
8	neutral	The person has a yellow ball going to the game.
	contradiction	The person is sitting in the bleachers.
	entailment	A person is playing with a ball.
147	neutral	A person is trying to get home from give a ball.
	contradiction	A person is reading a bank from london.
	entailment	A person is throwing a ball up.

z	Premise	Two women in bathing suits climb rock piles by the ocean.
2	neutral	Two women are climbing rocks in the ocean on a sunny day.
	contradiction	The women are playing basketball.
	entailment	Two women are climbing.
8	neutral	Two young women in bathing suits are friends
	contradiction	Two women naked.
	entailment	The girls looking at the water.
147	neutral	Two women are looking at the lagoon in front of a calm shore.
	contradiction	Two women are gossiping on a sandy beach.
	entailment	A group of women are climbing wood in the ocean.

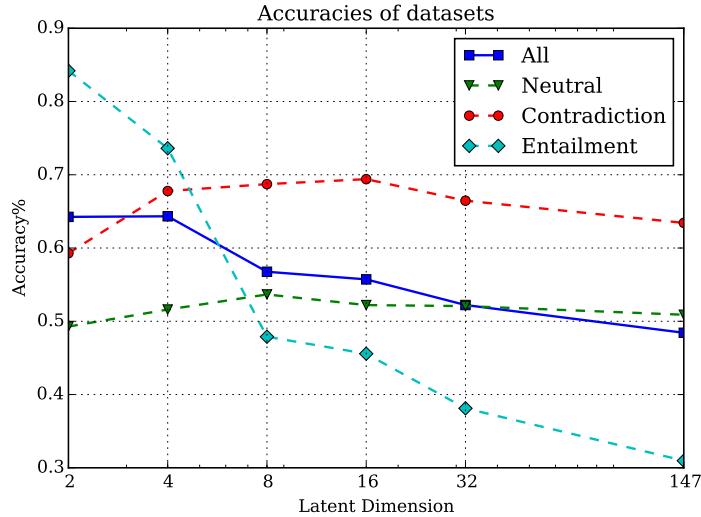
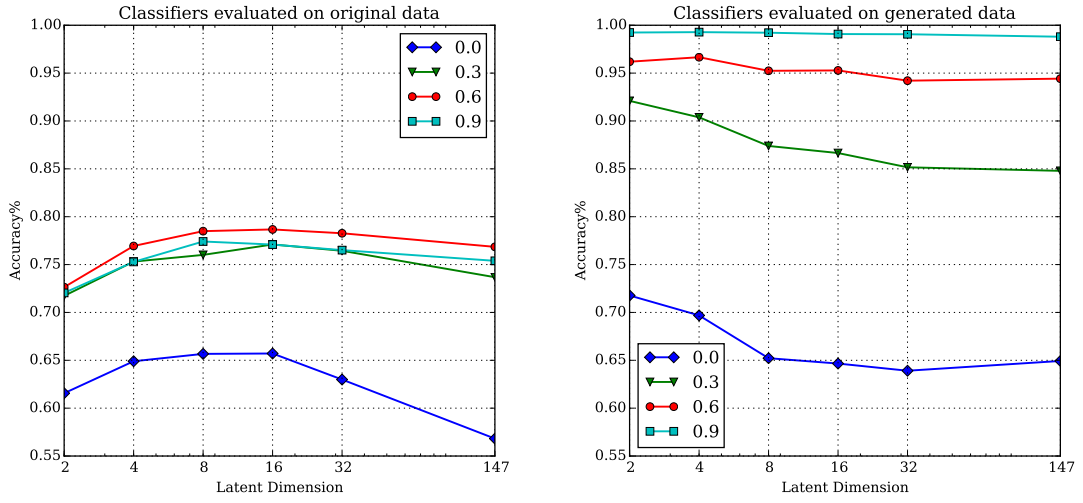


Figure 5.3: Accuracies of the unfiltered generated datasets classified by ORIGCLASS. A dataset was generated for each generative model with a different latent dimension $z \in [2, 4, 8, 16, 32, 147]$. For each dataset the examples were classified with ORIGCLASS. The predicted labels were taken as a golden truth and were compared to the labels of the generated dataset to measure its accuracy. The accuracies were measured for all the labels together and for each label separately.



(a) Accuracies of classifiers evaluated on the original test set (b) Accuracies of classifiers evaluated on generated development sets

Figure 5.4: Accuracies of classifiers trained on the generated dataset and tested on the original test set and the generated development sets. A dataset was generated for each generative model with different latent dimension $z \in [2, 4, 8, 16, 32, 147]$. From these unfiltered datasets new datasets were created by filtering according to various prediction thresholds (0.0, 0.3, 0.6, 0.9), which also represent chart lines. A classifier was trained on each of the datasets. Each point represents the accuracy of a single classifier. The classifiers were evaluated on the original test set in Figure 5.4a. Each classifier was evaluated on its corresponding generated development set in Figure 5.4b.

The unfiltered datasets have been evaluated with five other metrics besides classification accuracy. The results are presented in Figure 5.5. The whole figure shows the effect of latent dimensionality of the models on different metrics. The main purpose of the figure is not to show absolute values for each of the metrics, but to compare the metrics' curves to the curve of our main metric, the accuracy of the classifier.

The first metric – Premise-Hypothesis Distance – represents the average Jaccard distance between the premise and the generated hypothesis. Jaccard distance is defined as

$$d_J(P, H) = \frac{|P \cup H| - |P \cap H|}{|P \cup H|}, \quad (5.1)$$

where P represents the premise and H represents the hypothesis. Datasets generated with low latent dimensions have hypotheses more similar to premises, which indicates that the generated hypotheses are more trivial and less diverse than hypotheses generated with higher latent dimensions.

We also evaluated the models with standard language generation metrics ROUGE-L and METEOR. The metrics are anticorrelated with the accuracy of the classifier. We believe this is because the two metrics reward hypotheses that are similar to their reference (original) hypothesis. However, the classifier is better if trained on more diverse hypotheses.

The next metric is the log-likelihood of hypotheses in the development set. This metric is the negative of the training loss function. The log-likelihood improves with dimensionality since it is easier to fit the hypotheses in the training step having more dimensions. Consequently, the hypotheses in the generating step are more confident – they have lower log-likelihood.

The last metric – discriminative error rate – is calculated with the discriminative model. The model is trained on the hypotheses from the unfiltered generated dataset on one side and the original hypotheses on the other side. The error rate is calculated on the (generated and original) development sets. A higher error rate indicates that it is more difficult for the discriminative model to distinguish between the generated and the original hypotheses, which suggests that the original generating distribution and the distribution of the generative model are more similar. The discriminative model detects that low dimensional generative models generate more trivial examples as also indicated by the distance between the premise and hypotheses. On the other hand, it also detects the hypotheses of high dimensional models, which more frequently contain grammatic or semantic errors.

There is a positive correlation between the discriminative error rate and the accuracy of the classifier. This observation led us to the experiment, where the generated dataset was filtered according to the prediction probability of the discriminative model. Two disjoint filtered datasets were created. One with hypotheses that had high probability that they come from the original distribution and the other one with low probability. However, the accuracies of classifiers trained on these datasets were very similar to the accuracy of the classifier on the unfiltered dataset. A similar test was also done with the log-likelihood metric. The examples with the higher log-likelihood had a similar performance as the ones with the lower log-likelihood. This also led us to set the size of the beam to 1. Also, the run time of the generating hypothesis is $\mathcal{O}(b)$, where b is the beam size. Thus, with lower beam sizes much more hypotheses can be generated.

To accept the hypothesis from Section 4.6.1 we have shown that a quality dataset requires *accurate* examples by showing that filtering the dataset with the original classifier improves the performance (Figure 5.4a). Next, we have shown that *non-trivial* examples are also required. If the filtering threshold is set too high, these examples are excluded, and the accuracy drops. Also, the more trivial examples are produced by low-dimensional models, which is indicated by lower premise-hypothesis distances, and lower discriminative

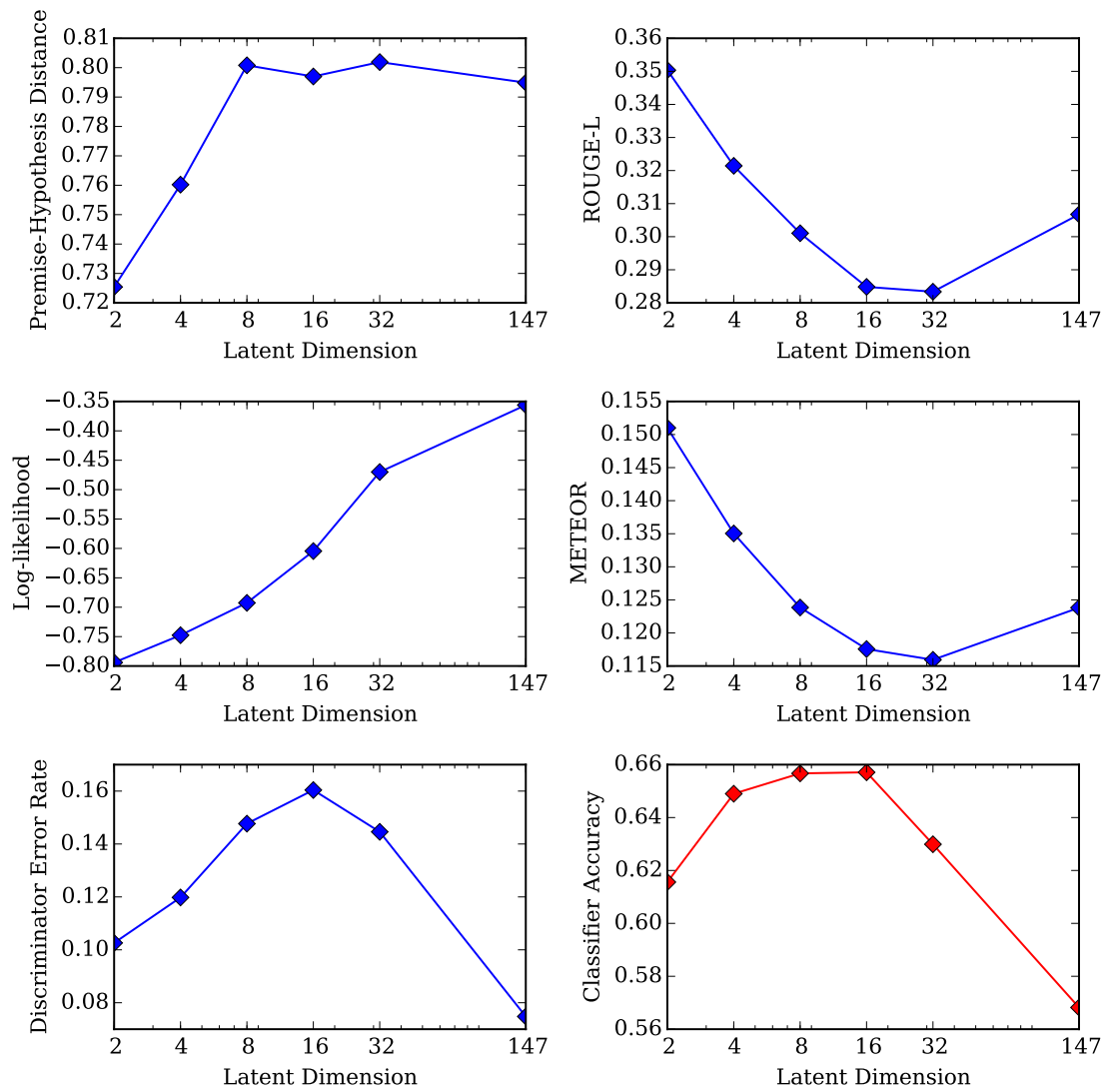


Figure 5.5: Comparison of unfiltered generated datasets (each generated by a model with a different latent dimension) using various metrics.

Table 5.12: Comparison of generative models. Column $|\theta_{total}|$ is the total number of trainable parameters. Column $|\theta_*|$ represents the number of parameters that are updated with each training example. Thus, hierarchical softmax and latent representation parameters are excluded from this measure. Columns $a@0.0$ and $a@0.6$ represent the accuracy of the classifier trained on the unfiltered dataset and on the dataset filtered with threshold 0.6, respectively. Column $a-dat$ presents the accuracy of the unfiltered development dataset evaluated by ORIGCLASS. Column nll presents the negative log-likelihood of the unfiltered development dataset. The error rates of the discriminative models are presented by der .

Model	d	$ \theta_{total} $	$ \theta_* $	a@0.0	a@0.6	a-dat	nll	der
ENCODERDECODER	150	6.4M	1.1M	43.4	72.4	57.5	1.00	0.01
VAEENCODERDECODER	150	6.4M	1.1M	58.6	77.9	48.0	0.77	1.9
BASEEMBEDDECODER	226	13M	580K	65.0	77.7	56.3	0.73	14.0
ATTEMBEDDECODER	150	11M	581K	65.7	78.5	56.8	0.69	14.8

error rate (Figure 5.5). Finally, a quality dataset requires more *comprehensible* examples. The high dimensional models produce less comprehensible hypotheses. They are detected by the discriminative model (see discriminator error rate in Figure 5.5).

5.4.3 Other models

We also compared ATTEMBEDDECODER model to all other models. Table 5.12 presents the results. For all the models the latent dimension z is set to 8, as it was previously shown to be one of the best dimensions.

For all the models the number of total parameters is relatively high, however only a portion of parameters get updated each time. The ATTEMBEDDECODER model was the best model according to our main metric – the accuracy of the classifier trained on the generated dataset.

The hidden dimension d of the BASEEMBEDDECODER was selected so that the model was comparable to ATTEMBEDDECODER in terms of the number of parameters θ_* . The accuracies of classifiers generated by BASEEMBEDDECODER are still lower than the accuracies of classifiers generated by ATTEMBEDDECODER, which shows that the attention mechanism helps the models.

Table 5.13 shows the performance of generated datasets compared to the original one. The best generated dataset was generated by ATTEMBEDDECODER. The accuracy of its classifier is only 2.7% lower than the accuracy of the classifier generated on the original human crafted dataset. The comparison of the best generated dataset to the original dataset shows that the datasets had only 0.06% of identical examples. The average length of the hypothesis was 7.97 and 8.19 in the original dataset and in the generated dataset, respectively. In another experiment the generated dataset and the original dataset were merged to train a new classifier. Thus, the merged dataset contained twice as many examples as other datasets. The accuracy of this classifier was 82.0%, which is 0.8 % better than the classifier trained solely on the original training set. However, the lowest average loss is achieved by the classifier trained on the original dataset.

5.4.4 Qualitative evaluation

We also did a qualitative evaluation of the generated hypothesis. Hypotheses are mostly grammatically sound. Sometimes the models incorrectly use indefinite articles, for instance “*an phone*”, or possessive pronouns “*a man uses her umbrella*”. These may be due

Table 5.13: The performance of classifiers trained on the original and generated datasets. The classifiers were tested on the original test set. The generated datasets were generated by the models from Table 5.12. The generated datasets were filtered with threshold 0.6.

Dataset	loss	accuracy%
ENCODERDECODER	1.563	72.4
VAEENCODERDECODER	1.174	77.9
BASEEMBEDDECODER	1.095	77.7
ATTEMBEDDECODER	0.970	78.5
Original Dataset	0.475	81.2
Original Dataset + ATTEMBEDDECODER	0.486	82.0

to the fact the system must learn the right indefinite article for every word separately. On the other hand, the models sometimes generate hypotheses that showcase more advanced grammatical patterns. For instance, hypothesis “*The man and woman have a cake for their family*” shows that the model can correctly use plural in a non-trivial setting. Generative neural networks have a tendency to repeat words, which sometimes make sentences meaningless, like “*A cup is drinking from a cup of coffee*” or even ungrammatical, like “*Several people in a car car*”.

As shown previously, the larger the latent dimension, the more creative hypotheses are generated. However, with increased creativity semantic errors emerge. Some hypotheses are correct, just unlikely to be written by a human, like “*A shirtless man is holding a guitar with a woman and a woman*”. Others present improbable events, like “*The girls were sitting in the park watching tv*”, or even impossible events, for instance “*The child is waiting for his wife*”. This type of errors arise because the models have not learned enough common sense logic. Finally, there are hypotheses, which make no sense. For instance, “*Two women with grassy beach has no tennis equipment*”. On the contrary, the models are able to generate some non-trivial hypotheses. From the original premise “*A band performing with a girl singing and a guy next to her singing as well while playing the guitar*”, the model has generated some hypotheses that do not contain concepts explicitly found in the premise. For instance, “*People are playing instruments*” (entailment), “*The band was entirely silent*” (contradiction), or “*The girl is playing at the concert*” (neutral).

Regarding the compliance of the hypotheses with the label and premise, we observed that many generated hypotheses are not complying with the label, however they would be a very good example with a different label. For instance, the generated hypotheses represent entailment instead of contradiction. This also explains why the accuracy of the generated dataset measured by the original classifier is low in Figure 5.3. On the other hand, the models generate examples that are more ambiguous and not as clear as those in the original dataset. These examples are harder to classify even for a human. For instance, the relationship between premise “*A kid hitting a baseball in a baseball field*” and hypothesis “*The baseball player is trying to get the ball*” can be either interpreted either as an entailment if verb *get* is interpreted as *not to miss* or contradiction if *get* is interpreted as *possess*. For a deeper insight into the generated hypothesis, more examples are presented in Appendix A.

The gap between the discriminative error rates (disc-er) of ENCODERDECODER models and EMBEDDECODER models in Table 5.12 is significant. To further investigate, the same experiment was performed again by a human evaluator and the discriminative model. This time on a sample of 200 examples. To recap, both the model and the human were asked to select the generated hypothesis given a random original and generated hypothesis without

Table 5.14: Discrimination error rate on the development set and a sample of 200 examples, evaluated by the discriminative model and human evaluator.

Gen. Model	Dev. set		Sample	
	Disc. Model	Human	Disc. Model	Human
ATTEMBEDDECODER	14.0	-	14.0	22.5
VAEENCODERDECODER	1.9	-	2.0	11.5

knowing which one is which.

Human evaluation confirms that ATTEMBEDDECODER hypotheses are more difficult to separate from the original one than the hypotheses of VAEENCODERDECODER. Table 5.14 presents the results. The discriminative model discriminates better than the human evaluator. This may be due to the fact that the discriminative model has learned from a large training set, while the human was not shown any training examples. Human evaluation has shown that generated hypotheses are positively recognized if they contain a grammatical or semantic error. But even if the generated hypothesis does not contain these errors, it sometimes reveals itself by not being as sophisticated as the original example. On the other hand, the discriminative model does not always recognize these discrepancies. It relies more on the differences in distributions learned from a big training set. The true number of non-distinguishable examples may be even higher than indicated by the human discriminator error rate since the human may have correctly guessed some of the examples he could not distinguish.

5.5 Approach Comparison

We compare the presented representations, learning approaches and the experimental results.

5.5.1 Text preprocessing

The first step of every approach is text preprocessing. In all the considered approaches, this includes sentence and token splitting. This step also includes assigning word embeddings to tokens. If an error occurs in this step, it might propagate throughout the whole process, and the final meaning representation could be wrong due to this error. If the approach learns from annotation layers, the error might occur in the process of generating annotations. For instance, we noticed that some person named-entities were tagged with organization type by named-entity recognizer, thus $\langle Person \rangle ::= \langle Organization \rangle$ context-free grammar rule was suggested. The rule was marked as *neutral* (not further used) by the human user to overcome this error. In another example, we noticed that some universities were not tagged as named-entities. For instance, *University of Arizona* was not recognized as organization named-entity, while *Arizona* was recognized as location. The grammar induction approach makes it possible to recognize the term as organization by suggesting the following rules: $\langle Organization \rangle ::= \langle Organization \rangle$ of $\langle Location \rangle$) and $\langle Organization \rangle ::= university$.

It is difficult to recognize and justifiably attribute the error to incorrect word vectors. However, in this respect we recognized that in NLI classification some examples containing similar word embeddings get misclassified. For instance, the classifier does not recognize that the color in premise is different than the color in the hypothesis, although the colors have similar word embeddings. For example, the premise “The man sits on a blue chair”

and hypothesis “The chair is red.” gets classified as entailment rather than contradiction.

5.5.2 Concept disambiguation

Another challenge with annotations is finding the correct concept in the ontology for a term. If the correct annotations are not provided in the dataset, like Wikipedia links in Section 5.3.1, a word sense disambiguation system should be used. To alleviate the word sense disambiguation problem in the evaluation of approaches that map directly to predicate logic (Section 5.1 and Section 5.2), we considered the mapping correct if one of the generated predicate logic relations was correct. However, we noticed that some patterns yield multiple relations (one for each combination of arguments) that need to be disambiguated. Table 5.3 shows that some rules had ambiguous matches (column *AmbigM*), which resulted in many more total relations (column *TotalRel*) than matches (column *Matches*). Word sense disambiguation can be implicitly done with contradiction inference. When a new relation is added to the ontology, the inference mechanism checks whether there are any contradictions. The contradictions may arise when a wrong concept is used to form the relation. In Section 5.2 the experiment shows that out of 326 sentences with corresponding logical relations 122 of them were not added to ontology due to contradictions. The task of word sense disambiguation is not only to select the correct concept but also recognize that the correct concept is missing in the ontology. Table 5.5 shows several scenarios, which happen when searching for denotations of phrases. In 58% of phrases (rows 2-5) the correct concept was missing. If there are other denotations for the term, then without word sense disambiguation an incorrect concept is selected (row 5). Otherwise, a new concept is generated. This concept is semantically poor because there are no other connections to the ontology. If some information exists from other layers, it can be attached to the new concept, like for instance named-entity type (row 2). In some cases (row 3) the term does not represent an atomic concept, and it can be split into smaller parts and then recombined into a compound concept. The grammar induction approach (Section 4.3) identifies such terms and suggests new rules to split the terms. An instance extraction method presented in Section 5.3.3 tries to identify terms that represent atomic concepts. The method is based on the grammar approach and also identifies the class of the concept, which makes the new concept semantically richer. In another experiment with Cyc’s ontology (Section 5.1) it is confirmed that although the majority of terms have corresponding concepts in the ontology, many of the concepts are still missing.

5.5.3 Sentence structure modeling

All the present approaches structure the sentence in a tree-like form, with an exception of recurrent neural networks, where the sentence is processed sequentially. In Section 5.2, we evaluate an approach based on XLE parser. The experiment shows that even though the dataset was filtered down to a group of sentences that are more suitable for the parser, there were still around a quarter of incorrect parses (see Table 5.4). On the positive side, the f-structures produced by the XLE parser contain additional information, like subcategorization frames that can be coupled with semantic resources from the ontology.

On the other hand, in the grammar induction approach the sentence structuring is learned in the process and is to some extent controlled by a human user. In the textual pattern-based approach, suggested textual patterns are generated by the n-gram approach, which does not consider the context tokens – tokens around the pattern. Many suggested patterns are thus not complete or they contain unnecessary tokens, like many patterns in Table 5.1. To reduce the amount of suggested erroneous patterns, several filtering metrics are calculated, like frequency, normalized expectation, number of arguments, etc.

In grammar induction, only frequency is considered when suggesting rules. It turned out that other metrics are not needed. However, for controlling the parsing a reliability measure has been proposed, where the trigger probability has a similar effect as the minimal token frequency in the pattern-based approach. Both metrics identify patterns that are too generic. Another advantage of the grammar approach is that there are less constraints imposed on the patterns. In the pattern-based approach universal tokens are not allowed on both ends of the pattern due to infeasibility to recognize arguments.

5.5.4 Cardinality

All the presented approaches that map to predicate logic are rule-based. As Figure 5.1a shows, the more rules that are generated, the more text is covered. In the grammar induction experiment, 252 non-neutral rules were generated to fully parse about a quarter of sentences, with a coverage of 78.5%. Figure 5.1a also shows that the growth of coverage and fully parsed sentences starts to slow down with the number of rules, which suggests that there is a long tail of rules needed to be generated to fully parse the dataset. In the pattern approach with syntax parser (Section 4.2) only 8 rules were created. However, some of these rules are coupled with numerous semantic resources, which effectively makes the number of rules higher. Also, some rules are very generic and frequently applied, but produce semantically poorer meaning representations, like *String instance rule* in Section 4.2.2. Such rules increase the coverage of the mapping process, however the quality of meaning representation becomes questionable. This implies that approaches should be evaluated by not only measuring the coverage of the mapping, but also the quality of meaning representations. Therefore, relation extraction models were trained on semantic trees to evaluate their quality. Similarly, classification models were trained on newly generated textual entailment examples to measure their quality. In both cases, better performance of a predictive model indicates better quality of the source, from which the model was trained on. While in predicate logic approaches the performance increases with the number of rules, the performance of textual entailment models increases with the number of training examples, as shown in (Bowman, Angeli, et al., 2015).

5.5.5 Manual work

All approaches except for NLI generation involve manual work to some degree. In the pattern-based approach, the textual pattern of the rule is suggested by the system, while the logic part of the rule is entirely constructed by the user. In the prototype presented in Chapter 6, the system also suggests concepts that the user can use to construct the logic pattern. Constructing logic patterns can be time consuming since the user needs good knowledge of the ontology or more time is spent for finding the right concepts. In the grammar induction approach, the rules are generated semi-automatically; the user only needs to validate the rules by assigning them the right property. In the NLI generation approach no extensive manual work has been done, however to increase the accuracy of the newly generated examples the user could validate them.

5.5.6 Meaning representations

In this dissertation, we work with three types of meaning representations: predicate logic, semantic trees and distributed representations. Predicate logic is the most explicit representation, where concepts from the ontology usually correspond to particular words or phrases in the text. The power of the representation or how well it models the meaning the text mainly depends on how the concepts are described and interconnected (there are lots

of relations and rules that connect them). One of the problems with using predicate logic for is missing vocabulary. Lots of concepts, especially predicates, needed to represent the text, are currently not existent in ontologies, such as Cyc, Freebase and DBpedia. To create and properly connect the missing concepts can be a difficult task since the meaning can be subjective and non-deterministic. We propose an approach for constructing a taxonomy from grammar (Section 4.4) and a method for discovering new instances (Section 5.3.3). In this line of work, we also propose semantic trees as meaning representation. Semantic trees are an intermediate representation between text and predicate logic. The recognized ontology concepts are represented by the leaf nodes, while other (unrecognized or non-existent) concepts are modeled either with inner nodes (these are represented by grammar rules) or null leaf nodes. Due to the tree structure of the representation it is easier to model minor variations in the text. For example, the automaton model in Figure 5.2 presents numerous ways of how the *institution* relation is expressed in text. Furthermore, every node of the tree is represented by a semantic class, which is itself placed in the ontology in the process of taxonomy extraction. Comparing to predicate logic representation semantic trees are more loosely defined. This makes them less rigid, thus they can be later utilized for a more goal-oriented task. For instance, they were used as a source for supervised learning of DBpedia relations.

On the other hand, we tailored the neural networks for generation of NLI hypotheses. The models are learned to distinguish between entailment, contradiction or neutral relation between two sentences, for which deep knowledge of the meaning of text is needed. To predict if hypothesis entails premise with predicate logic, the ontology is queried using the theorem prover with the query (*implies* #PREMISE #HYPOTHESIS), where #PREMISE and #HYPOTHESIS are relations representing the meaning of premise and hypothesis respectively. If the answer to the query is *True*, then the label of the example is *entailment*. Similarly, for detecting contradiction, one could use a query, like (*implies* #PREMISE (*not* #HYPOTHESIS)). Furthermore, if answers to both queries are not *True*⁸, then the label of the example is *neutral*. To correctly predict the following example: “The boy is riding an animal.” *entails* “The boy is riding a camel.”, a relation, like (*subClassOf* Camel Animal) is needed in the background knowledge. Therefore the recipe for solving NLI classification with predicate logic system is to map all the training examples with predicate logic, infer new rules and relations, then use them for prediction via querying. However, the textual entailment datasets may contain noisy examples. It would be very difficult to handle them with a non-probabilistic predicate logic approach, where a single rule or relation might mislead the prediction.

Generating hypothesis using predicate logic and ontology is more difficult. To generate hypotheses that entail a given premise, the query (*implies* #PREMISE ?HYPOTHESIS) (notice that the hypothesis is a variable) is executed against the ontology, then for each answer (relation) a hypothesis is generated using the module for generating a text from logic. But the query search space is much larger than in the classification task, since each valid relation represents one of the labels in respect to the premise.

⁸A typical inference engine would answer 'No results' or 'The query cannot be proven'.

Chapter 6

Prototype

This chapter presents a prototype for mapping text to predicate logic using textual patterns. The prototype is a web-based system that allows a human user to semi-automatically construct mapping rules presented in Section 4.1. Additionally, it allows a fast construction of relations that are the result of applying the newly created rules. The screenshot of the prototype is presented in Figure 6.1 and a video of the prototype is available at <https://www.youtube.com/watch?v=a75wqJTW4fM>.

The prototype has the following workflow. The user selects a piece of text, for which he wishes to construct a new rule. The pattern rule is created by generalizing the selected text, and the logical pattern is manually constructed using the suggestions from the ontology. Once the rule is created, the user can verify and add the relations produced by the rule to the ontology. Now, we present the prototype in more detail.

The process starts when the user selects a language and a document from that language. All the documents were preprocessed into the layered representation of text, presented in Section 3.1.1. In this particular version of the prototype there were several thousand documents from three languages available: English, Spanish and Catalan¹. The prototype is not limited to certain languages, however the quality of the process will be dependent on the ability to convert the documents into a good layered representation. Furthermore, if the language is not well connected with the ontology, it might take additional work.

Once selected, the document will appear on the right side, as it is shown in Figure 6.1. The user can then select a portion of text adequate for rule creation, and drags it into the *Lexical pattern* box. This triggers the interface for creating a new textual pattern, presented in Figure 6.2.

The user creates a textual pattern by selecting a layer for each token of selected text. In this prototype there were six possible layers: the original lexical layer, the lemmatized layer, two different parts of speech layers, a named entity layer, and a generic layer (*Other argument*). The generic layer token represents the universal token that can match any part of the text, as presented in Section 3.2.

Once the user confirms the textual pattern, the pattern will appear in the *Textual pattern* box, as it is shown in Figure 6.3. In another version of the prototype the patterns are suggested by the system with the method for discovering promising patterns presented in Section 4.1.1.2. The suggested patterns appear as orange highlighted parts of text, and if the user on one of the parts, the pattern of this part will appear in *Textual pattern* box. The frequency of the pattern is shown in the *Textual pattern* box, so that the user can see how many times the rule will be applied on the whole corpus. The next step is to create the logical part of the rule in the *Logical pattern* box. The system reads the pattern rule,

¹The particular languages were selected because the infrastructure for creating the layered representation was developed by XLike project (www.xlike.org).

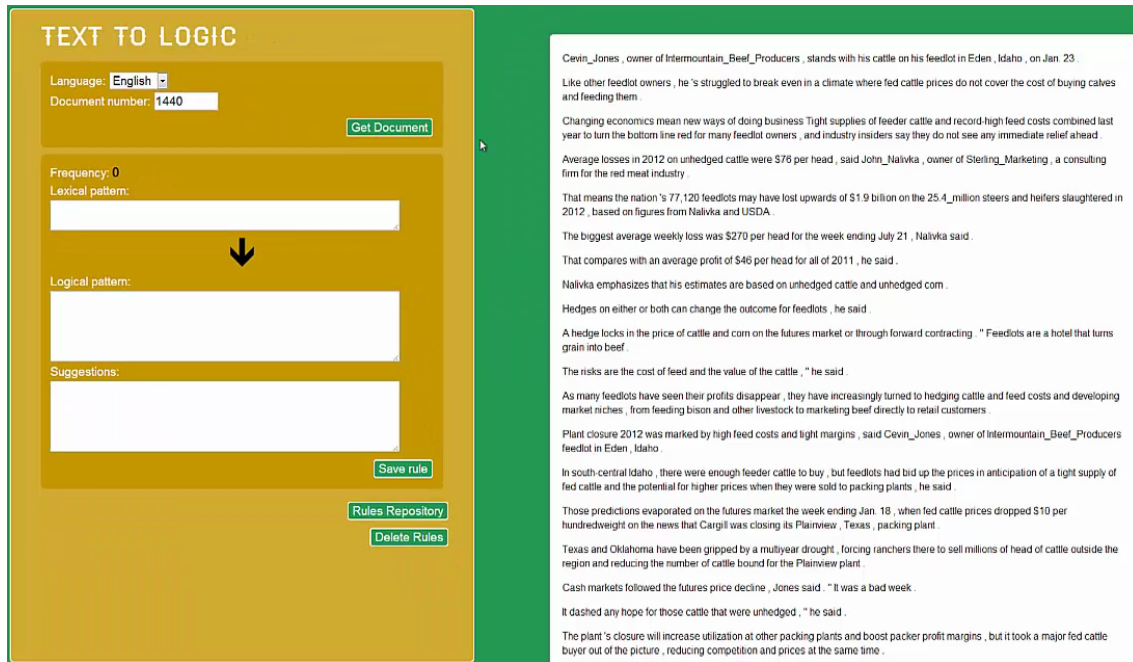


Figure 6.1: The screenshot of the main screen of the prototype.

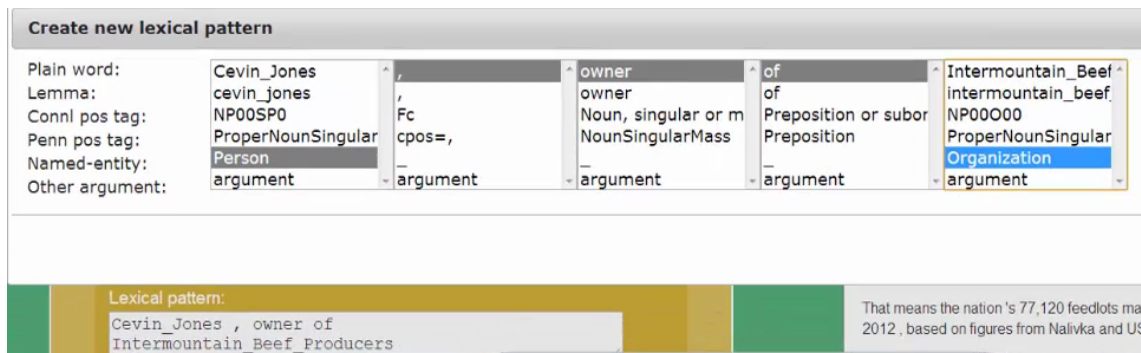


Figure 6.2: The screenshot of the interface for constructing lexical patterns.

and searches for the concepts in the ontology that are denoted by the lexical tokens in the textual pattern. These concepts are then presented to the user in the *Suggestions* box, so they can be used for constructing the logical pattern. For this prototype Cyc was used as the ontology. Once the user finishes the construction of the logical pattern, the rule is complete. When *Save Rule* is clicked, the rule is saved and applied on the whole corpus. The system searches for all parts of text that match the textual pattern in all documents and construct a relation for each of those parts using the logical pattern. The matching parts of text become orange in the displayed document.

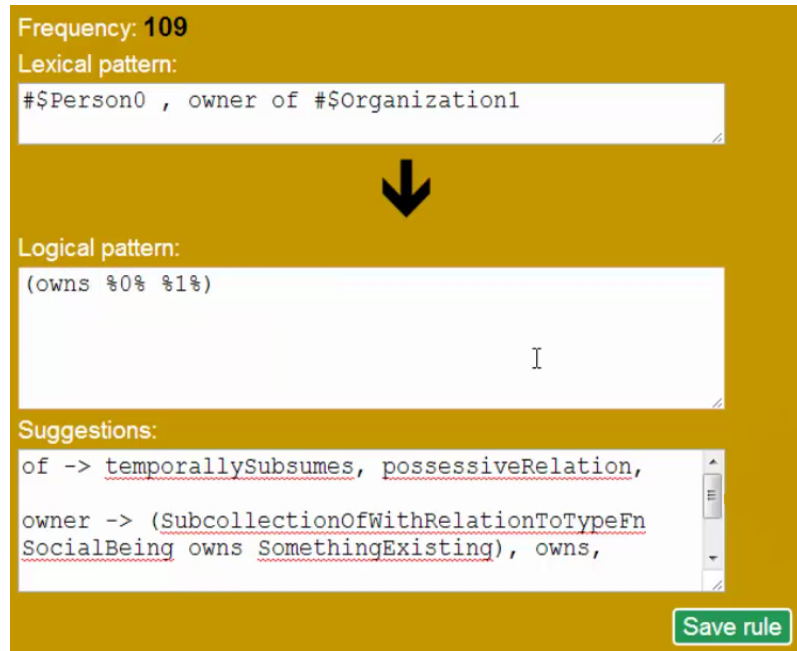


Figure 6.3: The screenshot of the interface for creating rules.

If the user clicks on one of them, the interface for adding the relation to the ontology will appear. This interface, presented in Figure 6.4, allows the user to confirm the proposed relation or change it by selecting other concepts as arguments. For each lexical argument, the system searches for corresponding concepts in the same way the logical pattern suggestions are searched for. If there are several corresponding concepts, then the user selects one of them, or creates a new one by clicking on the *Existing Constant* button to switch to *New Constant* mode. In this mode, the user defines the name of the new concept. If the lexical argument has no corresponding concepts in the ontology, the user must write in the name of the new concept (*New Constant* mode). The system suggests the new name using the tokens of the lexical argument. For instance, *CevinJones* is suggested for *Cevin Jones* in Figure 6.4. When the user is satisfied with the relation, it clicks the *Add* button to add the relation and possible new concepts to the ontology. This step is successful if there are no contradictions caused by the added relation. Once the user returns to the main interface (Figure 6.1), the part of the document for which the relation was successfully added becomes green.

The prototype also supports nested relations. For instance, for the sentence

Walter E. Williams is a professor of economics at George Mason University

the following relation can be constructed

```
(positionOfPersonInOrg WalterWilliams GeorgeMasonUniversity
  (ProfessorFn Economics))
```

The user first constructs compound arguments, like (ProfessorFn Economics) in the example. Compound concepts are created in the same way as relations, however they are not added to the ontology initially. They are added as an argument of the predicate relation, which is in our example the relation with predicate positionOfPersonInOrg in the example. Thus, if the user wants to create a relation with compound concepts (nested relation), all the compound concepts need to be created first.

Add to Knowledge Base

Document: 1440
Sentence: 0
LogicalPattern: (owns %0% %1%)

Cevin_Jones, **owner of Intermountain_Beef_Producers**, stands with his cattle on his feedlot in Eden , Idaho , on Jan. 23 .

Arguments:

Id	Lexical	Logical	New/Existing	Wrong Argument
1	Cevin Jones	CevinJones	New Constant	☐
2	Intermountain Beef Producers	IntermountainBeefPr	Existing Constant	☐

Logical formula: Assertible logical sentence

(owns CevinJones IntermountainBeefProducers)

Add

Figure 6.4: The screenshot of the interface for constructing and adding relations.

Chapter 7

Conclusions

In this dissertation we propose several approaches for text understanding using machine learning. We have evaluated and compared several meaning representations, including the proposed semantic trees, and learning approaches, which includes an approach for ontology learning using grammar induction, and an approach for generating NLI datasets.

With preliminary experimentation we have identified several limitations of approaches that map directly from text into predicate logic with mapping rules. The main limitations are incomplete ontology and excessive amount of manual work. To overcome them, we proposed an approach for joint learning of ontology and semantic parsing, which involves less manual work. The approach is based on the grammar induction procedure. The induced grammar is used to parse the sentences into semantic trees. The semantic trees are a meaning representation that models the sentence structure. Each node of the tree has a semantic class, and some of them are represented by ontology instances. On our person Wikipedia dataset we show that a few hundred grammar rules are enough to parse around 80% of the dataset and to fully parse a quarter of the sentences. Part-of-speech taggers or pretrained syntactic parsers are not needed for the approach, which makes it suitable for datasets and languages, for which such resources are unavailable. Semantic trees were further used to extract predicate logic representations and them to the existing ontology. In one task, the trees are used to discover rare instances in the dataset. The other task is relation extraction. We proposed and evaluated several learning models for relation extraction with various degrees of interpretability and performance. The other source for learning the ontology is the grammar. Classes, instances and their relations are extracted from the grammar rules to form the taxonomic part of the ontology.

The proposed ontology learning approach is very suitable for text describing facts, like the person Wikipedia datasets used in experiments. This dataset has lots of entity annotations and a fairly regular language. On the other hand, it may not be so suitable for less formal texts describing images, where more common sense knowledge is needed. One such dataset is SNLI, used for Natural Language Inference (NLI), where the goal is to find a logical relation between the premise and hypothesis. However, instead of solving the classification problem, we have been investigating the generative problem, where the goal is to generate the hypothesis given the premise and the label. The generative models also require reading comprehension, however they additionally require the capability to generate text. To evaluate the generative models we have proposed a new metric, the accuracy of the classifier trained on the generated dataset and tested on the original test set. The experimentation shows that this metric behaves differently than machine translation metrics ROUGE and METEOR, which are less favorable towards diversity. We proposed several generative neural networks for generating NLI datasets. The experimentation has shown that the best model achieves 78,5% accuracy, which is only 2.7 % less than the

accuracy of the classification model trained on the original dataset. Furthermore, the dataset combined of best generated and original dataset achieves the highest accuracy of 82%. Finally, on the best generated dataset the evaluation shows that the human in 22.5 % of cases incorrectly distinguishes between the generated and original hypothesis, which additionally displays the quality of the dataset. The best generative model contains a decoder with attention mechanism, and instead of the encoder it uses trainable mapping embeddings.

7.1 Scientific Contributions

The work presented in the dissertation contains the following contributions to the Natural Language Understanding:

- Context-free grammar induction approach that allows to build grammars, which are used to parse text into semantic trees and can be utilized by the proposed novel method for constructing the taxonomic part of the ontology (Sections 4.3 and 4.4). A grammar was induced on a first sentence Wikipedia person dataset, where a few hundred rules were induced and used to parse about a million sentences. More than hundred taxonomic relations were extracted from the grammar (Section 5.3.2).
- The proposed semantic trees coupled with the proposed ontology learning approaches: a relation extraction approach and an instance discovery approach (Section 4.5.2). Several relation extraction models were trained on semantic trees that were obtained by parsing the first sentence Wikipedia person dataset with the induced grammar. The models were used to predict 79 types of relations. The best model obtained 57.1 F1 score (Section 5.3.4).
- Procedure for evaluation of generated NLI text hypotheses. This includes a procedure for constructing a NLI dataset and an evaluation metric for the generated texts – the accuracy of the classifier trained on the generated dataset (Section 4.6). The metric supports generation of more diverse sentences compared to metrics ROUGE and METEOR. Furthermore, the metric rewards datasets that consist of accurate, non-trivial and comprehensive examples (Section 5.4.2).
- Text generative neural network, which consists of the decoder that learns a mapping embedding for each training example separately (Section 4.6.1.2). The attention-based version of this network achieves the best result measured by the proposed metric. The accuracy of the classifier trained on the generated dataset is only 2.7% lower than the classifier trained on the original dataset. While this generated dataset combined with the original dataset achieves the highest accuracy (Section 5.4.3).

7.2 Future Work

For future work, we envision several potential research directions for individual approaches as well as for the combination of approaches.

Semantic trees offer a compact tree-structured meaning representation, which could be exploited for scenarios not covered by this dissertation, like relation type discovery and question answering. Also, the grammar induction, from which semantic trees are generated, is not fully automatic. Machine learning techniques could be used instead of the human user for validation. Another idea for future work is to integrate some ideas from other grammar induction methods to detect meaningful patterns without relying on the

annotation of text. Furthermore, the grammar induction approach could also be adapted to other data modalities, such as images, speech or wherever data segmentation is feasible.

On the other hand, there are many possibilities for future work regarding the generation of NLI hypotheses. To obtain higher accuracies of the constructed datasets, they need to be filtered, because the generative models produce examples whose labels are not always accurate. Thus, we propose for future work incorporating the classifier into the generative model, in a similar fashion as it was done on images by (Lamb, Dumoulin, & Courville, 2016). Furthermore, the generative network could also include the discriminative model to generate examples from a distribution that is more similar to the original training distribution. We would also like to develop active learning methods to identify incorrect generated examples that would most improve the dataset if corrected.

The dissertation presents two seemingly orthogonal approaches to text understanding, the ontological and distributed representation approach. For future work, we envision combining the two approaches. For instance, using generative models to generate predicate logic instead of text. In this case, the input to the model is a sentence, while the output is its meaning expressed with predicate logic. Another potential approach is to learn the vector space embeddings of the ontology concepts. This will allow using ontological concepts from text as an input or output to a fully differentiable network, like our proposed generative model. Since none of the proposed approaches is heavily language dependent, we would like to use our models on languages other than English, and explore the possibilities for learning the mapping from several languages into a common meaning representation.

Appendix A

More Generated NLI Examples

In this section, more hypotheses generated by generative neural networks proposed in Section 4.6.1.2 are presented. Each example starts with the original example data. Then, several hypotheses generated from the original example with our best model are displayed.

Premise: a little boy in a blue shirt holding a toy.
Hypothesis: boy dressed in blue carrying toy to a toybox.
Label: neutral
Generated Hypotheses:
 the little boy is caucasian.
 a boy in blue holds a toy for the first time.
 a child in a blue shirt holds a toy outside.
 a little boy watches the toy to his friend.
 a little boy is playing with a toy in the kitchen.
 the boy likes to his toy.
 the boy is in the store.

Premise: a little boy in a blue shirt holding a toy.
Hypothesis: boy dressed in red lighting things on fire.
Label: contradiction
Generated Hypotheses:
 the little boy is dancing.
 a boy is jumping into the pool.
 a boy is watching television.
 the boy is sleeping on the floor.
 a boy is watching television indoors.
 a boy is holding a kitten.
 the little boy is standing in the pool.

Premise: a little boy in a blue shirt holding a toy.

Hypothesis: boy dressed in blue holds a toy.

Label: entailment

Generated Hypotheses:

the boy is holding a toy.

a kid is holding a toy.

a boy with a toy is riding a toy.

little little boy holding a toy.

a boy is riding on a toy.

a boy is with an object.

the little boy is wearing a shirt.

Premise: one woman with a dark coat on walking by herself in a city.

Hypothesis: the woman is walking on a sidewalk.

Label: neutral

Generated Hypotheses:

a woman at a city street has a sunny day.

the woman is shopping.

a woman is walking to the store.

the woman is asian.

the lady with the coat walking by her way to a store in the city.

the woman is afraid of the woman.

a woman was watching to go to a party

Premise: one woman with a dark coat on walking by herself in a city.

Hypothesis: the woman is standing in a building.

Label: contradiction

Generated Hypotheses:

the woman is sitting on a couch.

a woman is sitting on the ground.

a woman in a coat and a coat is sitting next to her way

a woman is watching the movie at the theater.

a man is walking on the street

the men are asleep

the dog is wearing a pink coat.

Premise: one woman with a dark coat on walking by herself in a city.

Hypothesis: the woman is outdoors.

Label: entailment

Generated Hypotheses:

a female is walking

there are people present.

woman walk.

a woman in a coat walking through a city.

there is a woman and a man in a crosswalk.

woman are walking by a city.

the woman is talking to each other.

Premise: two young boys of opposing teams play football, while wearing full protection uniforms and helmets.

Hypothesis: boys scoring a touchdown

Label: neutral

Generated Hypotheses:

the boys in uniform are trying to win the game.
 the brothers are wearing a uniforms for the football game.
 the boys were waiting to play the game.
 two boys are fighting with their team.
 two teams wearing uniforms being finished for the game
 two young boys are having fun.
 the boys are wearing helmets and are standing nearby.

Premise: two young boys of opposing teams play football, while wearing full protection uniforms and helmets.

Hypothesis: dog eats out of bowl

Label: contradiction

Generated Hypotheses:

the two boys are watching tv.
 the boys are playing baseball.
 the man is on a trampoline.
 two young girls are dressed in tuxedos.
 two kids playing with the ball at the beach.
 two kids are in the ocean.
 the boys are chasing an elephant

Premise: two young boys of opposing teams play football, while wearing full protection uniforms and helmets.

Hypothesis: boys play football

Label: entailment

Generated Hypotheses:

a trio of young boys wearing uniforms.
 two kids are being photographed for a football player.
 the boys have uniforms.
 two boys are wearing uniforms.
 the boys are dressed
 two boys are performing.
 the boys all in uniforms for their team.

References

- Aho, A. V. & Corasick, M. J. (1975, June). Efficient string matching: an aid to bibliographic search. *Commun. ACM*, 18(6), 333–340.
- Allen, J. (1995). *Natural language understanding (2nd ed.)* Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc.
- Artzi, Y., Lee, K., & Zettlemoyer, L. (2015). Broad-coverage ccg semantic parsing with amr. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 1699–1710).
- Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.
- Baker, C. F., Fillmore, C. J., & Lowe, J. B. (1998). The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1* (pp. 86–90). Association for Computational Linguistics.
- Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., ... Schneider, N. (2013, August). Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse* (pp. 178–186). Sofia, Bulgaria: Association for Computational Linguistics. Retrieved from <http://www.aclweb.org/anthology/W13-2322>
- Banchs, R. E. & Li, H. (2012). Iris: a chat-oriented dialogue system based on the vector space model. In *Proceedings of the ACL 2012 System Demonstrations* (pp. 37–42). Association for Computational Linguistics.
- Basile, V., Bos, J., Evang, K., & Venhuizen, N. (2012). Developing a large semantically annotated corpus. In *LREC* (Vol. 12, pp. 3196–3200).
- Belz, A. (2008). Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*, 14(04), 431–455.
- Berant, J., Chou, A., Frostig, R., & Liang, P. (2013). Semantic Parsing on Freebase from Question-Answer Pairs. In *EMNLP* (pp. 1533–1544).
- Bollacker, K., Evans, C., Paritosh, P., Sturge, T., & Taylor, J. (2008). Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data* (pp. 1247–1250).
- Bordes, A., Chopra, S., & Weston, J. (2014). Question answering with subgraph embeddings. *arXiv preprint arXiv:1406.3676*.
- Bos, J. & Markert, K. (2006). When logical inference helps determining textual entailment (and when it doesn't). In *Proceedings of the Second PASCAL RTE Challenge* (p. 26).
- Bowman, S. R., Angeli, G., Potts, C., & Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

- Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., & Bengio, S. (2015). Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.
- Brewster, C., Ciravegna, F., & Wilks, Y. (2002). User-centred ontology learning for knowledge management. In *Natural Language Processing and Information Systems* (pp. 203–207). Springer.
- Bunescu, R. C. & Mooney, R. J. (2005). A shortest path dependency kernel for relation extraction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing* (pp. 724–731).
- Cai, Q. & Yates, A. (2013). Large-scale Semantic Parsing via Schema Matching and Lexicon Extension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.
- Cambria, E. & White, B. (2014). Jumping nlp curves: a review of natural language processing research [review article]. *IEEE Computational Intelligence Magazine*, 9(2), 48–57.
- Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr, E. R., & Mitchell, T. M. (2010). Toward an architecture for never-ending language learning. In *AAAI* (Vol. 5, p. 3).
- Chen, D. & Manning, C. D. (2014). A fast and accurate dependency parser using neural networks. In *EMNLP* (pp. 740–750).
- Cheng, J., Dong, L., & Lapata, M. (2016). Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.
- Choi, E., Kwiatkowski, T., & Zettlemoyer, L. (2015, July). Scalable semantic parsing with partial ontologies. In *Proceedings of the 2015 Association for Computational Linguistics*.
- Chrupala, G. (2014). Normalizing tweets with edit scripts and recurrent neural embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics* (Vol. 2, pp. 680–686). Citeseer.
- Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., & Bengio, Y. (2015). A recurrent latent variable model for sequential data. In *Advances in neural information processing systems* (pp. 2980–2988).
- Clark, A. (2001). Unsupervised induction of stochastic context-free grammars using distributional clustering. In *Proceedings of the 2001 workshop on Computational Natural Language Learning-Volume 7* (p. 13).
- Clarke, J. & Lapata, M. (2008). Global inference for sentence compression: an integer linear programming approach. *Journal of Artificial Intelligence Research*, 31, 399–429.
- Collobert, R. & Weston, J. (2008). A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning* (pp. 160–167). ACM.
- Dagan, I., Glickman, O., & Magnini, B. (2006). The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising textual entailment* (pp. 177–190). Springer.
- Dahl, D. A., Bates, M., Brown, M., Fisher, W., Hunicke-Smith, K., Pallett, D., . . . Shriberg, E. (1994). Expanding the Scope of the ATIS Task: The ATIS-3 Corpus. In *Proceedings of the Workshop on Human Language Technology* (pp. 43–48).
- Das, D. & Smith, N. A. (2011). Semi-supervised frame-semantic parsing for unknown predicates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1* (pp. 1435–1444). Association for Computational Linguistics.

- Denkowski, M. & Lavie, A. (2014). Meteor universal: language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.
- Dias, G., Guilloré, S., & Lopes, J. (1999). Mutual expectation: a measure for multiword lexical unit extraction. In *Proceedings of VExTAL Venezia per il Trattamento Automatico delle Lingue*.
- D’Ulizia, A., Ferri, F., & Grifoni, P. (2011). A survey of grammatical inference methods for natural language learning. *Artificial Intelligence Review*, 36(1), 1–27.
- Elliott, D. & Keller, F. (2014). Comparing automatic evaluation measures for image description. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: Short Papers* (Vol. 452, p. 457).
- Etzioni, O., Banko, M., & Cafarella, M. J. (2006). Machine reading. In *AAAI* (Vol. 6, pp. 1517–1519).
- Falk, Y. N. (2011). *Lexical-functional grammar*. Oxford University Press.
- Fellbaum, C. (1998). *Wordnet*. Wiley Online Library.
- Fillmore, C. J. (1977). Scenes-and-frames semantics. *Linguistic structures processing*, 59, 55–88.
- Flanigan, J., Thomson, S., Carbonell, J., Dyer, C., & Smith, N. A. (2014). A discriminative graph-based parser for the abstract meaning representation. In *In ACL*. Citeseer.
- Frost, R. A. & Hafiz, R. (2006, May). A new top-down parsing algorithm to accommodate ambiguity and left recursion in polynomial time. *SIGPLAN Not.* 41(5), 46–54.
- Ge, R. & Mooney, R. J. (2005). A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the ninth conference on computational natural language learning* (pp. 9–16). Association for Computational Linguistics.
- Gildea, D. & Jurafsky, D. (2002). Automatic labeling of semantic roles. *Computational linguistics*, 28(3), 245–288.
- Glickman, O., Dagan, I., & Koppel, M. (2005). Web based probabilistic textual entailment. In *Proceedings of the 1st Pascal Challenge Workshop* (pp. 33–36).
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems* (pp. 2672–2680).
- Goodman, J. (2001). Classes for fast maximum entropy training. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, 2001* (Vol. 1, pp. 561–564). IEEE.
- Guha, R. V. & Lenat, D. B. (1991). Cyc: a mid-term report. *Applied Artificial Intelligence an International Journal*, 5(1), 45–86.
- Hänig, C., Bordag, S., & Quasthoff, U. (2008). UnsuParse: unsupervised Parsing with unsupervised Part of Speech Tagging. In *LREC*.
- Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735–1780.
- Hodosh, M., Young, P., & Hockenmaier, J. (2013). Framing image description as a ranking task: data, models and evaluation metrics. *Journal of Artificial Intelligence Research*, 47, 853–899.
- Irsoy, O. & Cardie, C. (2014). Opinion mining with deep recurrent neural networks. In *EMNLP* (pp. 720–728).
- Iyyer, M., Boyd-Graber, J. L., Claudino, L. M. B., Socher, R., & Daumé III, H. (2014). A neural network for factoid question answering over paragraphs. In *EMNLP* (pp. 633–644).

- Iyyer, M., Manjunatha, V., Boyd-Graber, J., & Daumé III, H. (2015). Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the Association for Computational Linguistics*.
- Johnson, R. & Zhang, T. (2014). Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*.
- Kamp, H. (1981). A theory of truth and semantic representation. *Formal semantics-the essential readings*, 189–222.
- Kingma, D. P. & Ba, J. (2014). Adam: a method for stochastic optimization. *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.
- Kingma, D. P. & Welling, M. (2013). Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)* (2014).
- Koehn, P. (2009). *Statistical machine translation*. Cambridge University Press.
- Kolesnyk, V., Rocktäschel, T., & Riedel, S. (2016). Generating natural language inference chains. *arXiv preprint arXiv:1606.01404*.
- Krishnamurthy, J. & Mitchell, T. M. (2015). Learning a compositional semantics for Freebase with an open predicate vocabulary. *Transactions of the Association for Computational Linguistics*, 3, 257–270.
- Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., & Steedman, M. (2011). Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (pp. 1512–1523).
- Lamb, A., Dumoulin, V., & Courville, A. (2016). Discriminative regularization for generative models. *arXiv preprint arXiv:1602.03220*.
- Lang, J. & Lapata, M. (2011). Unsupervised semantic role induction with graph partitioning. In *Proceedings of the conference on empirical methods in natural language processing* (pp. 1320–1331). Association for Computational Linguistics.
- Le, P. & Zuidema, W. (2014). The inside-outside recursive neural network model for dependency parsing. In *EMNLP* (pp. 729–739).
- Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., . . . Auer, S., et al. (2015). Dbpedia—a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2), 167–195.
- Lenat, D. B. (1995). Cyc: a large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11), 33–38.
- Li, J., Luong, T., & Jurafsky, D. (2015, July). A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (pp. 1106–1115). Beijing, China: Association for Computational Linguistics. Retrieved from <http://www.aclweb.org/anthology/P15-1107>
- Liang, P., Jordan, M. I., & Klein, D. (2013). Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2), 389–446.
- Lin, C.-Y. (2004). Rouge: a package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop* (Vol. 8). Barcelona, Spain.
- Liu, H. & Singh, P. (2004). Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4), 211–226.
- Liu, W., Weichselbraun, A., Scharl, A., & Chang, E. (2005). Semi-automatic ontology extension using spreading activation. *Journal of Universal Knowledge Management*, 1, 50–58.
- Lu, W., Ng, H. T., Lee, W. S., & Zettlemoyer, L. S. (2008). A generative model for parsing natural language to meaning representations. In *Proceedings of the Conference on*

- Empirical Methods in Natural Language Processing* (pp. 783–792). Association for Computational Linguistics.
- Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.
- Luong, M.-T., Sutskever, I., Le, Q. V., Vinyals, O., & Zaremba, W. (2015, July). Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (pp. 11–19). Beijing, China: Association for Computational Linguistics. Retrieved from <http://www.aclweb.org/anthology/P15-1002>
- MacCartney, B., Grenager, T., de Marneffe, M.-C., Cer, D., & Manning, C. D. (2006). Learning to recognize features of valid textual entailments. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics* (pp. 41–48). Association for Computational Linguistics.
- Mairesse, F., Gašić, M., Jurčiček, F., Keizer, S., Thomson, B., Yu, K., & Young, S. (2010). Phrase-based statistical language generation using graphical models and active learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics* (pp. 1552–1561). Association for Computational Linguistics.
- Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., & McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (pp. 55–60).
- Marcus, M. P., Marcinkiewicz, M. A., & Santorini, B. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2), 313–330.
- Maxwell, J. & Kaplan, R. (1996). An efficient parser for lfg. In *Proceedings of the First LFG Conference*.
- McGuinness, D. L., Van Harmelen, F. et al. (2004). Owl web ontology language overview. *W3C recommendation*, 10(10), 2004.
- Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In *Interspeech* (Vol. 2, p. 3).
- Mou, L., Men, R., Li, G., Xu, Y., Zhang, L., Yan, R., & Jin, Z. (2016). Natural language inference by tree-based convolution and heuristic matching. In *The 54th Annual Meeting of the Association for Computational Linguistics* (p. 130).
- Navigli, R. & Ponzetto, S. P. (2010). Babelnet: building a very large multilingual semantic network. In *Proceedings of the 48th annual meeting of the association for computational linguistics* (pp. 216–225). Association for Computational Linguistics.
- Navigli, R., Velardi, P., & Faralli, S. (2011). A graph-based algorithm for inducing lexical taxonomies from scratch. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three* (pp. 1872–1877). IJ-CAI'11.
- Norvig, P. (1991, March). Techniques for automatic memoization with applications to context-free parsing. *Comput. Linguist.* 17(1), 91–98.
- Cyc. (2016). Retrieved May 6, 2016, from <http://en.wikipedia.org/wiki/Cyc>
- DBpedia. (2016). Retrieved May 6, 2016, from <http://en.wikipedia.org/wiki/DBpedia>
- Freebase. (2016). Retrieved May 6, 2016, from <http://en.wikipedia.org/wiki/Freebase>
- Palmer, M., Gildea, D., & Kingsbury, P. (2005). The proposition bank: an annotated corpus of semantic roles. *Computational linguistics*, 31(1), 71–106.
- Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on as-*

- sociation for computational linguistics* (pp. 311–318). Association for Computational Linguistics.
- Parikh, A. P., Täckström, O., Das, D., & Uszkoreit, J. (2016). A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.
- Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: global vectors for word representation. In *EMNLP* (Vol. 14, pp. 1532–43).
- Poon, H. & Domingos, P. (2010). Unsupervised ontology induction from text. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics* (pp. 296–305). ACL '10.
- Pust, M., Hermjakob, U., Knight, K., Marcu, D., & May, J. (2015). Parsing english into abstract meaning representation using syntax-based machine translation. *Training*, 10, 218–021.
- Reddy, S., Lapata, M., & Steedman, M. (2014). Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2, 377–392.
- Reiter, E. & Belz, A. (2009). An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, 35(4), 529–558.
- Reiter, E., Robertson, R., & Osman, L. M. (2003). Lessons from a failure: generating tailored smoking cessation letters. *Artificial Intelligence*, 144(1), 41–58.
- Rocktäschel, T., Grefenstette, E., Hermann, K. M., Kočiský, T., & Blunsom, P. (2016). Reasoning about entailment with neural attention. In *International Conference on Learning Representations*.
- Rush, A. M., Chopra, S., & Weston, J. (2015, September). A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 379–389). Lisbon, Portugal: Association for Computational Linguistics. Retrieved from <https://aclweb.org/anthology/D/D15/D15-1044>
- Seginer, Y. (2007). Fast unsupervised incremental parsing. In *Annual Meeting-Association For Computational Linguistics* (Vol. 45, p. 384).
- Serban, I. V., Sordoni, A., Bengio, Y., Courville, A., & Pineau, J. (2016). Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*.
- Serban, I. V., Sordoni, A., Lowe, R., Charlin, L., Pineau, J., Courville, A., & Bengio, Y. (2016). A hierarchical latent variable encoder-decoder model for generating dialogues. *arXiv preprint arXiv:1605.06069*.
- Snow, R., Jurafsky, D., & Ng, Y. A. (2006). Semantic taxonomy induction from heterogeneous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics* (pp. 801–808).
- Socher, R., Bauer, J., Manning, C. D., & Ng, A. Y. (2013). Parsing with compositional vector grammars. In *ACL* (pp. 455–465).
- Socher, R., Karpathy, A., Le, Q. V., Manning, C. D., & Ng, A. Y. (2014). Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2, 207–218.
- Solan, Z., Horn, D., Ruppín, E., & Edelman, S. (2005). Unsupervised learning of natural languages. *Proceedings of the National Academy of Sciences of the United States of America*, 102(33), 11629–11634.
- Štajner, T., Rusu, D., Dali, L., Fortuna, B., Mladenović, D., & Grobelnik, M. (2010). A service oriented framework for natural language text enrichment. *Informatica*, 34(3).

- Starč, J. & Mladenović, D. (2016). Constructing a natural language inference dataset using generative neural networks. *In Review for Computer Speech and Language Journal (Impact Factor : 1.324)*.
- Starč, J. & Mladenović, D. (2017). Joint learning of ontology and semantic parser from text. *Intelligent Data Analysis Journal (Impact Factor : 0.631)*.
- Suchanek, F. M., Kasneci, G., & Weikum, G. (2007). Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web* (pp. 697–706). ACM.
- Sukhbaatar, S., Weston, J., Fergus, R., et al. (2015). End-to-end memory networks. In *Advances in neural information processing systems* (pp. 2440–2448).
- Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104–3112).
- Theano Development Team. (2016, May). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints, abs/1605.02688*. Retrieved from <http://arxiv.org/abs/1605.02688>
- Titov, I. & Klementiev, A. (2012). A bayesian approach to unsupervised semantic role induction. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics* (pp. 12–22). Association for Computational Linguistics.
- Trampuš, M. & Novak, B. (2012). Internals of an aggregated web news feed. In *Proceedings of the Fifteenth International Information Science Conference IS SiKDD 2012* (pp. 431–434).
- Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and tell: a neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3156–3164).
- Völker, J., Hitzler, P., & Cimiano, P. (2007). Acquisition of OWL DL axioms from lexical resources. In *The Semantic Web: Research and Applications* (pp. 670–685). Springer.
- Vrandečić, D. & Krötzsch, M. (2014). Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10), 78–85.
- Walter, S., Unger, C., & Cimiano, P. (2014). ATOLL—A framework for the automatic induction of ontology lexica. *Data & Knowledge Engineering*, 94, 148–162.
- Wang, S. & Jiang, J. (2016, June). Learning natural language inference with lstm. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 1442–1451). San Diego, California: Association for Computational Linguistics. Retrieved from <http://www.aclweb.org/anthology/N16-1170>
- Warren, D. H. (1981). Efficient processing of interactive relational data base queries expressed in logic. In *Proceedings of the seventh international conference on Very Large Data Bases-Volume 7* (pp. 272–281). VLDB Endowment.
- Watkinson, S. & Manandhar, S. (2001). A psychologically plausible and computationally effective approach to learning syntax. In *Proceedings of the 2001 workshop on Computational Natural Language Learning-Volume 7* (p. 20).
- Wen, T.-H., Gasic, M., Mrkšić, N., Su, P.-H., Vandyke, D., & Young, S. (2015, September). Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 1711–1721). Lisbon, Portugal: Association for Computational Linguistics. Retrieved from <http://aclweb.org/anthology/D15-1199>
- Wolfe, T., Dredze, M., Mayfield, J., McNamee, P., Harman, C., Finin, T., & Van Durme, B. (2015). Interactive knowledge base population. *arXiv preprint arXiv:1506.00301*.

- Wong, W., Liu, W., & Bennamoun, M. (2012). Ontology learning from text: a look back and into the future. *ACM Computing Surveys (CSUR)*, 44(4), 20.
- Wong, Y. W. & Mooney, R. J. (2006). Learning for semantic parsing with statistical machine translation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics* (pp. 439–446). Association for Computational Linguistics.
- Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., . . . Bengio, Y. (2015). Show, attend and tell: neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning* (pp. 2048–2057).
- Xue, N. & Palmer, M. (2004). Calibrating features for semantic role labeling. In *EMNLP* (pp. 88–94).
- Yan, X., Yang, J., Sohn, K., & Lee, H. (2015). Attribute2image: conditional image generation from visual attributes. *arXiv preprint arXiv:1512.00570*.
- Zelenko, D., Aone, C., & Richardella, A. (2003, March). Kernel methods for relation extraction. *J. Mach. Learn. Res.* 3, 1083–1106.
- Zelle, J. M. & Mooney, R. J. (1996). Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence* (pp. 1050–1055).
- Zouaq, A. & Nkambou, R. (2009, November). Evaluating the Generation of Domain Ontologies in the Knowledge Puzzle Project. *IEEE Trans. on Knowl. and Data Eng.* 21(11), 1559–1572.

Bibliography

Publications Related to the Thesis

Journal Articles

- Starč, J. & Mladenić, D. (2017). Joint learning of ontology and semantic parser from text. *Intelligent Data Analysis Journal (Impact Factor : 0.631)*.

Conference Paper

- Starč, J. & Fortuna, B. (2012a). Identifying good patterns for relation extraction. In *Proceedings of the 15th International Multiconference Information Society-IS* (pp. 205–208).
- Starč, J. & Fortuna, B. (2012b). Translating news to cycl using the xle parser. In *Proceedings of the 15th International Multiconference Information Society - IS 2012* (pp. 179–184).
- Starč, J. & Mladenić, D. (2013a). Semi-automatic construction of pattern rules for translation of natural language into semantic representation. In *5th Jožef Stefan International Postgraduate School Students Conference* (pp. 199–208).
- Starč, J. & Mladenić, D. (2013c). Semi-automatic rule construction for semantic linking of relation arguments. In *Proceedings of the 17th International Multiconference Information Society-IS* (pp. 155–158).

Other Publications

Conference Paper

- Bradeško, L., Fortuna, B., Mladenić, D., & Starč, J. (2011). Semantic technologies for personal information management. In *Information Technology Interfaces (ITI), Proceedings of the ITI 2011 33rd International Conference on* (pp. 187–192). IEEE.
- Bradeško, L., Starč, J., Mladenić, D., Grobelnik, M., & Witbrock, M. (2016). Curious cat conversational crowd based and context aware knowledge acquisition chat bot. In *Intelligent Systems (IS), 2016 IEEE 8th International Conference on* (pp. 239–252). IEEE.
- Bradesko, L., Starč, J., & Pacifico, S. (2015). Isaac bloomberg meets michael bloomberg: better entity disambiguation for the news. In *Proceedings of the 24th International Conference on World Wide Web* (pp. 631–635). ACM.
- Pacifico, S., Starč, J., Bradeško, L., Brank, J., & Grobelnik, M. (2016). Semantic annotation of text using open semantic resources. In *Encyclopedia of Machine Learning and Data Mining* (pp. 1–6).

Biography

Janez Starc was born in Ljubljana, Slovenia. He completed the Bachelor of Science degree in computer science and mathematics at the Faculty of Mathematics and the Physics and Faculty of Computer and Information Science, University of Ljubljana, in 2010. Afterwards, he was employed at the Artificial Intelligence Laboratory, Jožef Stefan Institute. As part of the Young Researchers programme he enrolled in the PhD programme Information and Communication Technologies at the Jožef Stefan International Postgraduate School. He collaborated on EU funded projects like XLike (Cross-lingual Knowledge Extraction). He was also working on the development of conversational crowd-based context-aware assistant Curious Cat. In 2014, he was an intern with Bloomberg L.P. (New York, USA), where he was working on entity disambiguation and knowledge infrastructure.