Bulgarian Academy of Sciences
Institute of Information and Communication Technologies

Alexander Nikolaev Popov

Modeling Lexical Knowledge for Natural Language Processing

Doctoral Thesis

Doctoral Program: Informatics

Professional Area: 4.6 Informatics and Computer Science

Supervisor: Kiril Simov

Sofia, 2018

Contents

Chapter 1

Introduction

1.1 Importance of the Topic

Modeling the lexicon of natural languages is a task with long-standing importance in a range of disciplines that study the principles through which linguistic systems enable communication among humans. Lexical knowledge has been given a cornerstone role in many models within theoretical linguistics. For instance, most of the formal grammars that attempt to explain the mechanisms of linguistic understanding and production rely heavily on the lexicon for coordinating the different levels at which meaning operates: syntax, semantics, discourse information. Lexical Functional Grammar (LFG; Dalrymple (2001)), Head-driven Phrase Structure Grammar (HPSG; Pollard & Sag (1994)), Conceptual Semantics (Jackendoff, 1992) – all of these formalisms and others attach a high degree of importance to meaning carried by lexical items and to the way they combine in more complex structures.

Computational linguistics and natural language processing (NLP) have inevitably followed theoretical linguistics in its attempt to represent lexical knowledge, be it for the purpose of building computational models that shed light on the operations of language or in order to solve practical problems through automated language processing systems. Even when no attempt is made to represent lexical knowledge explicitly, it is already implied in the formalisms that are utilized in those fields. For instance, syntactic parsers work with rigorously defined structures that are provided by linguistic theory and they also need training data annotated according to theoretical specifications. Already in those specifications there are built-in assumptions about the lexicon, even when they have to do merely with

the morphosyntactic restrictions that operate on words in sentences. But the lexicon can contain much richer information. Two different words with the same morphosyntactic characteristics can determine very different sentence or phrasal patterns: in terms of syntactic structure, semantic valency, admissible collocations, etc. Increasingly, it is lexicon entries that are seen as the locus of this type of information, which is utilized by numerous NLP applications performing tasks such as: word sense disambiguation (Navigli, 2009), entity linking (Moro, Raganato, & Navigli, 2014), machine translation (Vickrey et al., 2005), lexicalized syntactic/semantic parsing (Collins, 2003; Giuglea & Moschitti, 2006), word similarity calculation (Agirre, Alfonseca, et al., 2009), question answering (Moldovan & Rus, 2001), information retrieval (Stokoe et al., 2003), etc.

There are two broad approaches to modeling lexical knowledge in NLP, which are certainly not mutually exclusive and do get combined in some cases. One line of work attempts to formalize the lexicon according to a predefined framework. Such models, like HPSG, have the advantage that they provide descriptions which comply with certain specifications and can be used according to well-understood schemas: in order to fill in logical structures, make inference, check the well-formedness of linguistic expressions, etc. However, the detailed data structures that are used in formalisms of this kind are usually expensive and slow to build, especially when it comes to lexicons, which often need to specify tens of thousands of entries in order to be effective. There are other detailed symbolic models, apart from lexicalized grammar formalisms, that focus on other aspects of lexical knowledge. Such are, for instance, WordNet (Fellbaum, Christiane, 1998), FrameNet (Baker et al., 1998), VerbNet (Schuler, 2005), etc., i.e. resources that aim to describe the lexical semantic aspects of a language. Initiatives like WordNet have their origins in psycholinguistic research based on the assumptions that the relations between concepts and the overall semantic network structure of the lexicon are major factors for the construction of meaning in the human mind. The assumption has since been successfully explored in NLP and this is the reason for the continued development of resources of this kind across languages. With this kind of symbolic modeling, meaning is found in the relations that hold between lexical items, rather than in the entries for the items themselves. The creation and maintenance of such semantic networks is rather costly and time-consuming as well, but it provides yet another way to model interactions between words in natural language.

The second approach is oriented toward a more probabilistic and fuzzy representation of lexical information, as opposed to the categorical features and relations used in symbolic frameworks. Research initiatives such as Distributional Semantics seek to model the properties of linguistic items on the basis of when and how often these items are used. This is a data-intensive effort whose results are based on the analysis of large, representative corpora, so that good approximations of the actual distributions of the linguistic items can be obtained. Under this approach, models of meaning are induced automatically from the data, rather than through introspection and top-down designs on the part of the researcher. While the top-down way of lexical modeling offers rich and easily interpretable representations, for the creation of which no large amounts of data are needed, it is prone to overemphasizing or deemphasizing phenomena that are not covered by the limited data, in addition to being sometimes prohibitively expensive to apply in actual analysis (due to the need for large annotation efforts involving trained experts).

The two rough approaches outlined above can clearly complement each other in different ways. Some linguistic structure is certainly necessary in order to explain the more complex relations that arise in natural language and to go beyond mere statistical co-occurrence. But this kind of theoretical analysis can be fruitfully augmented by information that is hard to formalize efficiently (in the sense of writing rules and formal descriptions): statistical counts of syntactic structure distribution, collocation patterns, degree of distributional similarity, etc. Moreover, there already are methods to generalize higher-order linguistic principles from raw data and incorporate them back into theoretical descriptions. The same applies in the other direction as well – theoretical structures can be used to produce data for the purpose of obtaining distributional models. Since language is a system that can be analyzed on multiple levels and since those separate levels of analysis interact strongly with each other, it is only natural that lexical information encoded in words and phrases should be rich, structured, diverse and context-sensitive. Thus, combining theoretical and statistical views of the nature of lexical knowledge is a path worth exploring and likely to yield valuable results. Models that succeed in capturing the wealth of linguistic information carried by the lexicon can help in automatically modeling linguistic structure at multiple levels. Such knowledge in turn should allow for better applications that rely on text analysis, more advanced feature generation for machine learning systems, perhaps even theoretical insights based on big data analytics.

1.2 Goals and Tasks of the Thesis

This work was undertaken in conjunction with a project for the construction of a Bulgarian version of the WordNet lexical resource and for its use for performing word sense disambiguation in running text. That work was partly situated in a project for building machine translation (MT) systems incorporating deeper linguistic analytical capabilities. Since the MT systems under development were covering both the Bulgarian-to-English and English-to-Bulgarian directions of translation, the research was focused on both languages in parallel. This has led, in the course of my work, to engaging with linguistic resources in both languages, and with such resources that are more often than not related to WordNet. Thus, modeling lexical knowledge in the dissertation is very much assuming the WordNet model as a starting point.

The general goals of the dissertation are the following:

- To explore WordNet as the basis for solving lexical analysis tasks for NLP and, if possible, to enrich its semantic model.
- To explore different automatic methods for solving one primary lexical analysis task – word sense disambiguation (WSD).
- To develop distributional models of lexical knowledge that also incorporate knowledge derived from theory.
- To explore different aspects of lexical knowledge, how they can be encoded and what is the interaction between them (e.g. when performing in parallel several kinds of lexical analyses on the same piece of text).

These overarching goals are reflected in the following list of focused tasks that I attempt to solve in this thesis:

1. Enhancing the WordNet semantic network for the purpose of knowledge-based word sense disambiguation.
2. Designing and implementing a neural network architecture for sequence-to-sequence annotation.

3. Applying the neural architecture to the task of part-of-speech tagging; experimenting with distributional morphosyntactic models as a source of features for learning.

4. Adapting the neural architecture for word sense disambiguation.

5. Deriving different distributional lexical models based on enrichments of the WordNet semantic network; testing the models on tasks such as knowledge-based and supervised WSD and similarity/relatedness calculation.

6. Implementing neural systems for multi-task learning in order to test whether and to what extent different aspects of lexical knowledge interact with each other.

1.3 Structure of the Thesis

The dissertation is structured as follows. There is an introduction, followed by seven chapters and a conclusion. It consists of 145 pages. The introduction outlines the topic and the main goals and contributions of the dissertation. It is followed by a chapter that defines formally the relevant tasks and by another chapter that provides an overview of previous work on the same and related topics. The following five chapters constitute the main part of the dissertation. Chapter 4 describes a solution to the task of POS tagging. Chapter 5 presents work on enriching the structure of a semantic network resource (WordNet) and evaluating the new networks with regards to knowledge-based WSD. Chapter 6 extends this line of work to training distributed representation models of words and word senses on the basis of such semantic networks. Chapter 7 presents two alternative solutions to the task of WSD. Chapter 8 explores multi-task learning with hybrid architectures that combine the previously discussed tasks. The conclusion briefly recapitulates the methods for modeling lexical knowledge covered in the preceding chapters, as well as the significant results that are reported. It also contains: a list of the author's publications related to the thesis; information about academic presentations and projects related to the approbation of the results; key contributions and outlook to future work. A declaration of originality of the results is enclosed, followed by a list of the tables, a list of the figures and a bibliography of the referenced works. Sources are cited in the thesis.

Chapter 2

Problem Definition

This chapter provides definitions of the problems set out before the thesis. Each of the tasks presented serves as a testing ground for the hypotheses explored in this work, as they are directly related to questions of representing lexical knowledge about linguistic items. The problems are first described in a non-formal manner, together with intuitive examples for illustration. Then a formal definition is given, and finally the specific linguistic resources in use are discussed, as those play a major role in constraining the problem definition (due to practical circumstances, such as dictionary size, training data coverage, etc.). This paves the way for the next chapter, which introduces important related work on automatic lexical analysis in NLP.

2.1 Part-of-Speech Tagging

One of the most general characteristics of lexical items is their *part of speech* classification. Lexical items grouped under the same part of speech category share important commonalities: they are likely to appear in the same positions in particular syntactic structures, they are often morphologically alike (in terms of their constituents and the grammatical features they bear), they also tend to share some relatively abstract conceptual features (e.g. nouns tend to refer to things, or when not doing so, to present processes, events and other phenomena as more *thing-like*).

There is no single way to organize the lexicon of a language into POS categories. Different criteria have been devised by linguists and languages themselves differ

in their POS repertoires. Moreover, the shape of the lexicon is also determined by the level of granularity at which POS categories are analyzed – a more fine-grained approach will yield a greater number of POS tags (e.g. verbs in a language can be analyzed into separate categories, such as auxiliary, modal, etc., or lumped together in one broad category "VERB"; further sophistication can be introduced by adding grammatical categories, such as person, tense, number, to the definition of POS categories). In the context of NLP, deciding what POS categories to include in a lexicon and how granular they should be depends largely on what that information will be used for. POS tagging often is configured as one task along a pipeline of processing modules, each dependent on the preceding ones. Syntactic parsers, for instance, rely strongly on POS tags, regardless of whether they are based on probabilistic rules of classification techniques. In such an application, fine-grained POS tags can be central markers of what a correct analysis would look like (for instance, information about grammatical categories like person and number can signal coordination between words occurring far from each other in a sentence). In other applications, however, more coarse-grained POS analysis might be sufficient – in WSD, to give an example that is relevant to this work, too fine-grained distinctions between POS tags might in fact hinder generalization.

Regardless of how granular the analysis is, there is one crucial distinction that obtains in all available POS tag sets. This distinction is between closed class (also called "function" or "grammatical") words and open class words (sometimes called "content" or "lexical" words). The former (determiners, prepositions, conjunctions, etc.) admit new members very rarely and express abstract and/or highly grammaticalized concepts, while the latter (nouns, verbs, adjectives, adverbs) categories are renewed constantly, with new members being added or falling out of use all the time. This aspect of the lexicon is important with regards to another of the major tasks under investigation in this work – word sense disambiguation. Closed class words typically exhibit a lower degree of ambiguity, unlike open class words. It is often the case that a word form may be ambiguous not simply with regards to the possible word senses it can select, but also with regards to the ways it can be analyzed in terms of POS categories. For instance, "tan" in English may be used as a noun, verb or adjective. Here are three of its possible uses (the glosses are supplied from WordNet): (noun) a browning of the skin resulting from exposure to the rays of the sun; (verb) treat skins and hides with tannic acid so as to convert them into leather; (adjective) of a light yellowish-brown color. This kind of ambiguity means that accurate POS tagging is of major importance for

the subsequent correct disambiguation of word senses and illustrates how separate aspects of lexical knowledge interact with each other (in this case, POS category and conceptual meaning). Therefore, one could argue that a coarser POS tag set would be preferable when the tags are to be used by a WSD system, since the coarseness is likely to lead to higher precision of the POS annotation.

2.1.1 Formal Definition

Let us give the task a formal specification. By POS tagging we will mean the procedure that assigns a label $p_i \in P$ to each word in a text T. Or in other words, finding a mapping A such that:

$$A(w_i) = p_j$$

where:

- $w_i \in T = (w_1, w_2, ..., w_n)$
- $p_j \in P$

T is the text that is to be POS tagged (hence it is important that the words in it appear in a sequence) and P is the tag set from which the POS labels are drawn. We want A to give us a label for each and every one of the words in T.

2.1.2 Data sets

As already discussed, there can be big variations between different POS tag sets. This work is going to present results on data for the Bulgarian language, more specifically the BulTreeBank POS tagged corpus (Simov & Osenova, 2004), using a medium-coarse-grained tag set of 153 labels (the finest-grained tag set has 680 labels, though only 581 are attested in the corpus; Simov et al. (2004)). The corpus contains about 38,000 POS-tagged sentences. In the final part of the dissertation, where WSD and POS are solved in parallel, I use the SemCor corpus for English (Miller et al., 1993). That corpus is a subset of the Brown corpus, which is tagged with a 87-label tag set (Kucera & Francis, 1982); a simpler tag set of just 12 POS categories is used for that experiment (Petrov et al., 2011).

2.2 Word Sense Disambiguation

Ambiguity is ubiquitous in human languages. It is found at multiple levels of analysis, but perhaps it is most pronounced with regards to the lexical senses that words select when used in context. As already discussed in the previous section, a great number of the open class words in a language exhibit a degree of lexical ambiguity. For instance, consider these sentences:

(1) The bat is a nocturnal animal.

(2) He swung the bat with all of his strength.

The two objects referred to in (1) and (2) are clearly very different. That the same word form is used to refer to them both is a coincidence in the language. Humans are very good at making such distinctions and can very effectively pick on clues from the surrounding context. In the case of (1) and (2) probably even a relatively simple algorithm that has access to good dictionary descriptions of the separate senses of the word can distinguish between the two (e.g. by counting how many of the words in the context appear in the dictionary definition of a particular sense, which is in practice the popular *Lesk algorithm* (Lesk, 1986)). But lexical ambiguity comes in many forms:

(3) The cows have been grazing much better and gaining weight lately.

(4) Three pairs of humpback cows and calves were spotted off the shore last month.

(5) He cows his opponents with displays of rhetorical mastery.

The word form *cows* in these sentences bears three distinct meanings: (3) mature female of mammals of which the male is called 'bull'; (4) the female of certain other large animals, for example elephant, rhinoceros, whale, or seal; (5) cause (someone) to submit to one's wishes by intimidation[1]. Even if we were to lemmatize the text, that is analyze the word forms to their dictionary forms, that would not make the lexical distinction apparent on the surface (the lemma of all instances of *cows* above is *cow*). It would be of help if we could successfully perform POS tagging, which would identify the instances in (3) and (4) as nouns and the instance in (5) as a verb. And since there is only one word sense of *cow* as a verb (namely, to scare somebody into submission), that method would disambiguate at least one of the uses.

[1]Source: https://en.oxforddictionaries.com/definition/cow

But distinctions such as that between the instances in (3) and (4) cannot be dealt away with so easily. Moreover, here the difference in meaning is more subtle than that demonstrated in (1) and (2); the senses in (3) and (4) are actually polysemously related, rather than just coincidentally sharing the same surface expression. More concretely, the sense in (4) seems to be a generalization of that in (3), but in order to infer that the more general sense is in use, one has to have access to multiple bits of knowledge: to what species of animals are *cow* and *calf* applicable, what are the natural habitats of these animals (as opposed to that of oxen), which of them are typically "spotted", what does "humpback" mean, etc. Questions of granularity are central to WSD as well, one would argue even more so than to POS tagging. The word sense in (3), for instance, can be subsumed under the one in (4) and that would facilitate the task significantly, but there would also be a degree of information loss as well.

As briefly demonstrated, language-theoretical issues are central to the definition of the WSD task. There are thus multiple ways distinctions in meaning can be defined and motivated, at varying magnitudes. This work assumes that word senses can be identified as separate linguistic entities and grouped together with relation to word forms – into dictionaries or other resources. There are lines of research in semi-supervised WSD wherein senses are inferred from data, but this approach does not provide a common lexical framework for investigating the lexicon and therefore I am not going to touch upon it in the thesis.

2.2.1 Formal Definition

Formally, by WSD we will mean the task of finding a mapping A such that[2]:

$$A(w_i) = s^j_{w_i}$$

where:

- $w_i \in T = (w_1, w_2, ..., w_n)$
- $s^j_{w_i} \in (s^1_{w_i}, s^2_{w_i}, ..., s^k_{w_i}) = Senses_D(w_i)$
- $D = (Senses_D(w_1), Senses_D(w_2), ..., Senses_D(w_l))$

[2]This formalization follows relatively closely Navigli (2009).

As with POS tagging, T is the text we are working with. However, for each word in T that has an entry in a dictionary D there can be k distinct corresponding word senses, with k not being fixed to a specific positive number. That is, in WSD there is no single tag set which is used for classifying all the words in the text, but each word found in the dictionary has its own tag set. This makes WSD in fact a collection of classification tasks. Note that here we assume that A selects only one pertinent word sense, but a system can be configured to select multiple senses, with different or identical degrees of certainty.

2.2.2 Variants of the WSD task

There are two widely adopted variants of the WSD task, which is reflected in the popular Senseval/SemEval competitions. One is the *lexical sample* WSD task, where a system is required to disambiguate a specific subset of the open class words per sentence (typically just one of them). The other variant is the *all-words* WSD task, where all open class words (or those that appear in a dictionary) are the target of disambiguation. As a demonstration of the difference between the two, let us look at one of the earlier examples once again:

(6) He cows his opponents with displays of rhetorical mastery.

In the lexical sample task only one of the words would be selected for disambiguation. It could be any of the open class words, but let us say that "displays" is that word. In WN, the lemma "display" is linked to 6 nominal senses and 2 verbal ones. A system solving this task would then need to select one of these options for this particular context. Typically, such systems are trained against separate data sets per lemma – each sentence in them contains word sense annotations only of that particular lemma and nothing else. With the all-words task, on the other hand, all open class words in the sentence would typically be selected for classification: "cows", "opponents", "displays", "rhetorical", "mastery".

The first variant is easier with regards to gathering training and evaluation data, since much less human annotation needs to be done in preparation. The all-words task requires much larger amounts of data, since systems are expected to work with the full lexicon (or an approximation of it). This work will focus entirely on the all-words task – because it is more challenging and because most downstream applications that rely on the output of WSD systems need to get this kind of lexical information about all open class words.

2.2.3 Data sets

The datasets that I use are as follows. For training supervised systems: SemCor (Miller et al., 1993) is used for training on English data, it contains roughly 360,000 words from the Brown corpus, about 226,000 of which are tagged with WordNet senses (Miller et al., 1993). For development and evaluation, the following datasets are used: Senseval-2 (2283 sense annotations of nouns, verbs, adjectives and adverbs; Edmonds & Cotton (2001)), Senseval-3 (1850 annotations of nouns, verbs, adjectives and adverbs from editorial, news story and fiction texts; Snyder & Palmer (2004)), SemEval-2007 (455 annotations of verbs and nouns only; Pradhan et al. (2007), SemEval-2013 (1644 annotations of nouns only; Navigli et al. (2013)), SemEval-2015 (1022 annotations of nouns, verbs, adjectives and adverbs from biomedical, math/computer science, social studies; Moro & Navigli (2015)). In the case of knowledge-based WSD, the models are evaluated on SemCor for English data and on the BulTreeBank corpus for Bulgarian data (Popov et al., 2014)[3]. In one case the SemEval-2013 data is also used for evaluation.

2.3 Word Similarity and Relatedness

I use this final task in the thesis in order to evaluate the quality of the distributional lexical models that are generated. Calculating the semantic similarity and relatedness of pairs of words amounts to determining how close in meaning those words are. It is important to bear in mind that this closeness is here interpreted on the basis of taking the words alone and comparing them only with regards to their lexical semantics, i.e. no contextual information of particular usages is taken into account. Another important note is that there is a significant difference between "similarity" and "relatedness". The former measures to what extent two terms mean the same thing, while the latter express to what degree the terms can be expected to occur in the same context.

Calculating such scores is a highly subjective endeavor, therefore the very measures that are employed to this purpose are to some extent arbitrarily chosen. Typically, however, a numerical scale of closeness is selected when constructing gold resources (e.g. from 1 to 10) and the annotators evaluate how alike or related the members of word pairs are. In the thesis the rankings of word pairs calculated

[3]Popov, A., Kancheva, S., Manova, S., Radev, I., Simov, K., & Osenova, P. (2014). The Sense Annotation of Bultreebank. Proceedings of TLT13, 127-136.

by automated systems are related to the gold rankings using Spearman's rank correlation.

2.3.1 Datasets

The following datasets are used for evaluating the performance of distributed lexical models on word similarity and relatedness calculation: WordSim-353 Relatedness, WordSim-353 Similarity (Agirre, Alfonseca, et al. (2009) describe the two subsets of the WordSim-353 data set; Finkelstein et al. (2001) describes the original data set) and SimLex-999 (Hill et al., 2015). The first one measures relatedness, while the latter two are used to evaluate similarity calculation. Models of the lexicon that perform better on one or the other datasets will obviously be characterized by specific features (e.g. better grouping of synonyms and antonyms, or better clustering of related concepts).

Chapter 3

Background and Related Work

This chapter presents previous work that is relevant to the dissertation thesis, both as motivation for it and as direct influence on the ideas explored. First, several explicit models of the lexicon are discussed – resources that aim to organize knowledge about lexical items in a structured manner, so as to be applicable to different kinds of NLP and, more generally, computational tasks. In particular, one of these resources – WordNet – serves as a chief object of investigation and research tool of the thesis. Having introduced it, the overview then moves on to discuss different approaches to WSD. Particular focus is given to knowledge-based WSD, as it is one of the methods employed in this work. The section that follows provides a brief overview of neural network approaches to NLP and then pays special attention to neural network language models, whose use has become nearly ubiquitous in the last few years, and to neural network methods for sequence labeling – both tasks of particular importance to the thesis. Finally, multi-task learning is briefly touched upon.

3.1 Explicit Models of the Lexicon[1]

3.1.1 Word Sense Inventories

As discussed in the problem definition section, doing WSD and being able to evaluate results and compare them fairly among systems depends on an available enumeration of the possible words senses for the words in the lexicon. Settling

[1] As with the problem definition part dedicated to WSD, this section largely follows Navigli (2009).

on a suitable enumeration methodology is in itself a big theoretical issue. There are many difficult points to be considered: how granular should the partition of meaning into word senses be (i.e. should senses be more or less general), how domain-centered or domain-independent an enumeration is desired, how should lexical change (the addition, loss and shift in meaning of word senses over time) be handled, etc.

There have been attempts to design lexicons which capture meaning as structures of semantic components and can assemble senses dynamically from the deeper representation of lexical items. One of the most popular examples of such projects is the *Generative Lexicon* developed by James Pustejovsky (Pustejovsky, 1991). Lexical items, under this generative approach, are characterized by a lexical type, argument structure, event structure and, most importantly to the lexical qualification, four *qualia*. The latter are akin to semantic features, which together provide a description of the item, respectively: formal (the category that most expressively characterizes the item in a larger domain), constitutive (how the parts of the item are related to it), telic (what is the purpose or function of the thing denoted by the lexical item) and agentive (factors involved in the origin or "bringing about" of an object) qualia (Pustejovsky, 1995). Thus structured, lexical knowledge can be dynamically constructed in context and interfaced more naturally with syntactic structures. A somewhat similar methodology to describing the lexicon that provides compositional capabilities and aims to express the meaning of lexical items through semantic primitives is Conceptual Semantics (Jackendoff, 1992).

An investigation into how compositional lexicons can be integrated with enumerative lexicons is certainly an important task, but this work is going to focus on an analysis that uses enumerative resources only (although it will become clear that a way to carry out such an incorporation might be implicitly present in the methodologies investigated). To this purpose, the most popular computational resource with enumerated senses is selected as lexicon – WordNet (Fellbaum, Christiane, 1998).

3.1.2 WordNet

Word senses in WordNet (WN) are organized in *synonym sets* or *synsets*. A synset consists of one or more word senses that are used to refer to roughly the same concept. Each word sense is a pairing of a lemma and a concept; the latter is

shared among the word senses grouped in a synset. Naturally, slight differences in meaning between the separate word senses do exist, as there are no absolute synonyms in natural language; but even though their usage might differ somewhat, the core meaning of the associated concept is shared among the synonyms. If a synset consists of more than one entry, those are in effect the same concept, but expressed through different words, which all share the same POS category. A word, or rather the lemma of a word in the lexicon, can have multiple senses associated with it and they will all be members of different synsets. For instance, here are the nominal senses of the word "school" taken from the online interface to WordNet, version 3.1[2]:

$$\begin{aligned}\text{Senses}_{\text{WN}}(\text{school}_\text{n}) = \{&\{\text{school}_\text{n}{}^{1}\},\\ &\{\text{school}_\text{n}{}^{2}, \text{schoolhouse}_\text{n}{}^{1}\},\\ &\{\text{school}_\text{n}{}^{3}, \text{schooling}_\text{n}{}^{2}\},\\ &\{\text{school}_\text{n}{}^{4}\},\\ &\{\text{school}_\text{n}{}^{5}, \text{schooltime}_\text{n}{}^{1}, \text{school day}_\text{n}{}^{2}\},\\ &\{\text{school}_\text{n}{}^{6}\},\\ &\{\text{school}_\text{n}{}^{7}, \text{shoal}_\text{n}{}^{3}\}\}\end{aligned}$$

As can be seen, some of its senses constitute synsets by themselves, like $school_n{}^1$, whose gloss in WN is "an educational institution", $school_n{}^4$ ("a body of creative artists or writers or thinkers linked by a similar style or by similar teachers") or $school_n{}^6$ ("an educational institution's faculty and students"). The senses that are grouped in synsets together with senses of other words refer to concepts whose meaning should be clear from the groupings; e.g. the second sense of "school" refers to a building, its fifth refers to a period of time and its last one – to a group of fish. The senses for the word are ranked by lexicographers according to the primacy of their use, which is also an important piece of information, as will become clear.

In this dissertation I assume WordNet 3.0 as the lexicon. This version contains sense mappings for a little over 147,000 strings. Each string may be associated with senses that bear any of the four open class POS labels: {n, v, r, a} or {noun, verb, adverb, adjective}. The total number of synsets in the lexicon is about 117,000 and the total number of POS-associated strings is around 155,000. The different POS are characterized on the whole with different degrees of polysemy.

[2]http://wordnetweb.princeton.edu/perl/webwn

Thus, for instance, nouns have a degree of polysemy 1.24, if monosemous strings are counted, and 2.79, if monosemous strings are excluded[3].

Synsets also contain: a short definition (called *gloss*), just like the ones provided for the different senses of "school"; some example sentences; unique identifiers; frequency counts (calculated against a sense-tagged corpus) and information about membership in the so called "lexicographer files" (also called "supersense tags") – groupings of senses defined on the basis of syntactic and logical properties, such as "noun.artifact", "verb.change", etc.

Finally, synsets and word senses are connected with each other through a set of basic lexico-semantic relations. Those relations are all binary and what makes them so important is that they in practice connect the otherwise isolated entries of WN into a semantic network, or semantic graph. This structure provides another view at lexical meaning: lexical items can now be situated in a web of concepts and thus their relative position in it becomes a strong descriptor of meaning. WordNet was in fact developed as a resource for testing psycholinguistic hypotheses and thanks to this structure it does indeed mimic some aspects of how lexical knowledge is supposedly stored in the human brain. Two types of relations are provided in WN: lexical and semantic. The lexical ones connect specific word senses (i.e. the senses of specific words that might be grouped together in a synset with other senses), while the semantic relations connect whole synsets (i.e. these relations apply to all members of the two connected synsets). Here I list the types of relations, as these will be important at a later stage of the dissertation.

Lexical relations in WN:

- Antonymy: When two senses are used to express the opposite meaning, e.g. $hot_a{}^1$ is the opposite of $cold_a{}^1$.
- Pertainymy: Obtains between an adjective and a noun that the adjective can be said "to pertain to", e.g. $feline_a{}^1$ pertains to $cat_n{}^1$.
- Derivational relation: When one word form is in fact derived from another, such as in nominalization ($dive_n{}^2$ nominalizes $dive_v{}^2$), verbalization ($water_v{}^1$ verbalizes $water_n{}^1$), and others (e.g. $slowness_n{}^1$ *is derived from* $slow_a{}^3$).

Semantic relations in WN:

- Hypernymy: A taxonomic relation which expresses the fact that A is a kind of B (this is also called an *is-a* relation), e.g. $lion_n{}^1$ is a kind of $big\_cat_n{}^1$, i.e.

[3]For more information see `https://wordnet.princeton.edu/documentation/wnstats7wn`

the latter sense is a *hypernym* of the former (*hyper* means "over" in Greek). This relation holds between nominal and verbal synsets.

- Hyponymy: The inverse relation of hypernymy, but applicable only to nominal synsets, e.g. $lion_n{}^1$ has as one of its hyponyms $lioness_n{}^1$.
- Troponymy: The inverse relation of hypernymy, but applicable only to verbal synsets, e.g. $communicate_v{}^1$ has as one of its troponyms $send\_a\_message_v{}^1$.
- Meronymy: A relation denoting that B is a part of A (also called a *part-of* relation), e.g. $beak_n{}^2$ is a meronym of $bird_n{}^1$.
- Holonymy: The inverse of meronymy, i.e. A has a part – B, e.g. $computer_n{}^1$ is a holonym of $processor_n{}^3$.
- Entailment: When the activity (or process, state, etc.) expressed by one verb automatically implies that another activity expressed by another verb is also true, e.g. $walk_v{}^1$ entails $step_v{}^1$.
- Similarity: This relation obtains between adjectives that are similar in meaning to each other, e.g. $slow_a{}^1$ is similar to $sluggish_a{}^1$.
- Attribute: When a certain value regarding a noun is expressed via an adjective (usually in a relative manner), e.g. $slow_a{}^1$ expresses the value of the attribute $speed_n{}^2$.
- See also: This relation indicates that two adjectives are related, e.g. $slow_a{}^1$ is similar to $unhurried_a{}^1$.

Though first developed for somewhat different purposes, WordNet has evolved into the de facto standard for computational semantic lexicons in NLP. Different projects have sprung up over the years: for annotating corpora with senses from the WN inventory, developing WordNet versions in languages other than English[4], consolidating multilingual WordNet indices (Bond et al., 2016), etc. For better or for worse, the sense inventory of WN (or modifications of it) is almost universally used in WSD tasks, competitions, evaluation frameworks. This work is no exception – it assumes the Princeton WordNet 3.0 as a lexicon for the English word senses; for the Bulgarian word senses it employs a partial inventory mapped to the original WordNet hierarchy and used to further annotate a treebank (Popov et al., 2014).

[4]http://globalwordnet.org/wordnets-in-the-world/

3.1.3 Other Knowledge Resources

In addition to an enumeration of word senses, WSD systems often benefit from the use of additional knowledge resources. The interaction between conceptual bits of knowledge, as assembled in specific, context-dependent communicative situations, is highly complex and dependent on multiple structuring systems, some of which are extra-linguistic in nature. In fact, modern research suggests that meaning is always, to a differing degree, constructed dynamically and creatively, operating within the restraints of body and world rather than merely manipulating learned definitions of units of meaning (Bergen, 2012). Therefore, linking together different resources that can provide complementary representations of lexical knowledge and learning how to leverage that interconnected web of resources is a central task in computational lexical analysis.

There are many such resources that are of interest, but covering them is outside of the scope of this work. A few of them need to be singled out, however, since they are directly related to it. Most important of them is SemCor (Miller et al., 1993), a semantically annotated portion of the Brown Corpus (Kučera & Francis, 1967). SemCor is the largest sense-annotated corpus available, with 226,040 sense annotations spread over 352 documents and based on the WN sense inventory (another large annotated corpus is OntoNotes (Weischedel et al., 2013), but it uses a somewhat modified version of the WN sense groupings). It is the resource most widely used for training WSD systems. This work too uses SemCor for the training of supervised systems; in addition to that, it also uses it for enriching semantic knowledge graphs used for knowledge-based WSD and for evaluating the accuracy of the systems. Another resource used is *eXtended WordNet* (Mihalcea & Moldovan, 2001), a project to annotate the WN glosses with WN senses, so that the annotated text is available as training/testing data or as some kind of knowledge enhancement of WN.

Among the many other available knowledge resources, several more will mentioned here in order to give a sense of the possible lines of further work regarding the issue of integrating dense conceptual representations of lexical knowledge into a single model. VerbNet (Schuler, 2005) is a resource that organizes English verbs by mapping PropBank (Kingsbury & Palmer, 2002) verb types to Levin verb classes (Levin, 1993)[5]. It provides an inheritance hierarchy of the verb senses, where every

[5]In her seminal study Levin provides a classification of over 3,000 English verbs on the basis of their semantics and syntactic behavior. The commonalities between verbs are used to group

class and subclass is presented together with the syntactic and semantic structures it can be a predicate of. The verb senses are also mapped to other lexical resources such as WordNet and FrameNet via the SemLink project[6]. FrameNet (Baker et al., 1998) is a resource built on the principles of Frame Semantics, a theory that seeks to organize lexical knowledge in an encyclopedic manner, into *frames*, or structured collections of facts that are used to guide procedural knowledge. For instance, the "buying" frame informs the meaning of the word "buy" via knowledge about the different actors, objects, attributes, etc. that are or can be brought together in a buying scenario. FrameNet uses *semantic roles* like VerbNet, but its semantic roles are much more conceptually specific (e.g. the frame for buying has its specific roles, such as "Buyer", "Goods", "Means", etc.).

All of these lexical resources provide parts of the whole picture – by combining information about lexical items that has to do with their function at the levels of lexical and sentential semantics, syntax, conceptual representation, etc. Knowledge resources such as Wikipedia and DBpedia also provide important information that is complementary – encyclopedic knowledge about the world and the entities that inhabit it. Bringing all these separate resources together is an important task and in fact there is already work that has attempted to link together some of them (such as the BabelNet[7] project that brings together multiple resources including WordNet, Wikipedia, VerbNet, FrameNet, Open Multilingual WordNet, GeoNames, etc.; see Navigli & Ponzetto (2012) for more information).

3.2 Word Sense Disambiguation – an Overview

Word sense disambiguation, as defined in the previous chapter, is one of the most popular NLP tasks having to do with lexical and semantic analysis, and also one of the most difficult in NLP and artificial intelligence in general. Some of the reasons for its difficulty have been touched upon already: it combines in itself multiple classification tasks; the number of senses per word vary in number (from 1-2 up to dozens); depending on the lexicon chosen, the senses might be too granular to make clear distinctions between them; training data is sparse and insufficient to cover all words and senses; the interaction between linguistic units

them in a coherent hierarchy of verb types. PropBank is a corpus in which verb predicates and their arguments are annotated with abstract arguments that are akin to semantic roles like "Agent", "Theme", etc.

[6] http://verbs.colorado.edu/semlink/

[7] http://babelnet.org/

is so multi-faceted and dynamic that very rich and diverse knowledge resources are needed to perform correct disambiguation; etc. This section provides a brief look at some of the most popular approaches to solving the WSD task and their respective advantages and shortcomings. As noted in the previous chapter, only WSD with respect to words whose senses are readily enumerable (via a lexicon of some sort) is taken into consideration. Therefore, no references to unsupervised learning methods are provided in what follows. Focus is put on supervised and knowledge-based methods, which are an object of study in the dissertation. A presentation of supervised methods using neural networks is postponed until the next section, which deals exclusively with neural network approaches in NLP.

3.2.1 Supervised Methods for WSD

Supervised machine learning (ML) methods for WSD are aimed at training classifiers that can label words in text, with their senses available in an enumerative resource. The classifiers are trained on labeled corpora of natural language text, where some or all open class words (depending on the lexicon used and on the type of WSD task) have been disambiguated manually. Different supervised learning algorithms can be trained on such data. All of them depend crucially on an input representation of the data that is to be labeled. Typically in NLP tasks this representation is constructed as a set of features: bits of information that describe the data through a particular theory-informed filter. The usual repertoire of features for supervised ML models for NLP includes: the string of the token[8] considered for classification as well as of those surrounding it (this context is constrained within a window of observation and the tokens do not have to be those directly adjacent to the central one, but can be obtained by other methods, e.g. following its syntactic dependencies in the sentence), combinations of the string tokens, the lemmas of the tokens, their POS tags, the syntactic and semantic roles of the analyzed token, etc. More global features can be included as well: a domain, genre or topic label for the text as a whole, a list of topics or entities mentioned in it, etc. Thus, depending on what features the models takes in, the input data may have to go through a number of preprocessing steps.

[8]Tokens in NLP are the individual language units that are analyzed (in isolation or in context). Typically, text is tokenized into words, punctuation marks, numbers and other relatively autonomous symbolic units, before more complex analyses can be carried out. This is also the case with WSD.

Navigli (2009) provides an overview of several supervised algorithms. *Decision lists* and *decision trees* are two options for learning relatively compact and human-interpretable models for WSD. These algorithms leverage the training data in order to learn a set of rules capturing dependencies relevant to the correct labeling of tokens. These rules are expressed using the feature representations of the context, e.g. what string is the target word followed by, what POS categories, what tokens is it preceded by, in what positions precisely, etc. Under the list approach, the rules are given scores depending on how powerful they are and the highest scoring one is selected. Under the tree approach, rules are organized in a binary-branching structure, so that each binary choice against a particular rule narrows down the space of available answers; when a terminal node in the tree for a particular word is reached, a single sense can be selected. Even though these approaches offer the advantages of succinctness and interpretability, they suffer from issues arising from data sparseness and the inflexibility of symbolic rules.

Another algorithm presented there is Naive Bayes, which relies on calculating the conditional probabilities of senses $\{s_1(w), \ldots, s_n(w)\}$ for word w when a particular set of features is observed in the context. The probabilities for each feature are calculated from the training data. This algorithm assumes that the different features are independent of one another, which is too often a wrong assumption with regards to language. Even so, Naive Bayes classification performs respectably across different NLP tasks and with regards to WSD in particular, and it is fast to estimate.

Exemplar-based learning is yet another relatively robust and easy to estimate kind of model. The algorithm looks at the training data and keeps in memory the observed training instances as points in the feature space. New instances are then situated in the same space and classification is carried out depending on which sense cluster they are closest to. Simple methods such as the *k-Nearest Neighbor* algorithm are used to estimate closeness to the different clusters.

The system that consistently achieves the highest score or is among the highest scoring contenders on WSD tasks at the moment (see for instance Raganato, Camacho-Collados, & Navigli (2017)) is a supervised one – It Makes Sense (IMS), a support vector machine (SVM) algorithm with a fine-tuned feature-extraction step (Zhong & Ng, 2010). A previous version of IMS took part in the SemEval-2007 competition where it ranked among the best WSD systems (Pradhan et al., 2007). Since most of the top-ranked systems at this point were supervised ones

(among the SemEval-2007 top systems were also Maximum Entropy, Naive Bayes and k-Nearest Neighbor classifiers), but very few of those were open source and available for easy configuration and evaluation, IMS was developed precisely with such features in mind. It offers a default pipeline of pre-processing tools (tokenizer, POS tagger, lemmatizer), in which users can integrate their own tools, as well as a module for easily configuring the feature extractor. The default classification algorithm is LIBLINEAR (Fan et al., 2008) with a linear kernel. Since then, IMS has been evaluated with different feature extractors, including with an integration of word embeddings as features, with very good results (Taghipour & Ng, 2015; Iacobacci et al., 2016). Papandrea et al. (2017) demonstrate *SupWSD* – a Java API for WSD that seeks to replicate all the advantages offered by IMS, but also provides a software kit for WSD that is easy to use, configure and extend. The evaluation of the tool with the best-performing settings reported for IMS consistently matches up the original results or even surpasses them in some cases.

3.2.2 Knowledge-based Word Sense Disambiguation

Knowledge-based WSD (KBWSD) is a family of methods for approaching the task which do not rely on statistical knowledge learned directly from data, but rather exploit information that is encoded in resources modeling the lexicon. Thus, such methods are not directly data-driven, they are theory-driven (however, insofar as theory is always informed by data, it can be said that they are, indirectly, data-driven as well). A big advantage of KBWSD is that the algorithms of this broad family have full coverage over the senses in the lexicon that is used. Since the algorithms rely only on the information in the same knowledge resources that provide the enumeration of senses, this in theory allows them to make informed choices regarding each and every disambiguation case, in contrast to supervised approaches, which are tied to the availability of wide-coverage training data. The downside to KBWSD is that it rarely achieves results rivaling those of supervised systems. In addition to that, as was discussed in the section on knowledge resources, human language lexicons coordinate all kinds of data at various levels of linguistic analysis and the composition of lexical items is often very complex and creative. Therefore, defining a lexicon/knowledge resource that can exhaustively and effectively capture lexical knowledge for the purposes of KBWSD is a tremendous hurdle, both theoretically and practically.

The different KBWSD approaches explored in the literature all reflect the availability of resources and the ways in which the information encoded in them is put to use. In the absence of large structured lexical resources, early work in WSD was focused on simpler strategies for using lexical knowledge. One of the earliest and simplest methods for KBWSD relies on calculating the degree of overlap between target word definitions. This algorithm, called *the Lesk algorithm* after its inventor (Lesk, 1986), takes the dictionary definitions of word senses for target words that occur together in a shared context and calculates the degree of overlap between them. For two words w_1 and w_2, those two senses – ${s_{w_1}}^j$ and ${s_{w_2}}^k$ – are selected that share the highest number of words in their respective definitions. This calculation is manageable when just two words are considered, but with larger contexts the number of combinations to be considered grows exponentially, since the final selection of senses has to satisfy the above condition globally over the whole context. There exist variants of the algorithm which significantly minimize its complexity, e.g. by calculating the overlap between target word sense definitions and the words in the actual context only. Other modifications of the Lesk algorithm expand the definitions of target words by adding the definitions of the words contained in the first-level glosses, or by adding definitions of other concepts connected to the original sense via the structure of a semantic network such as WordNet (Banerjee & Pedersen, 2003). Oele & van Noord (2018) combine the Lesk algorithm with word embeddings and obtain improvements over the original version[9]. However, at present approaches based on definitions overlap are generally not able to achieve state-of-the-art results among KBWSD systems.

The construction of computational lexicons like WordNet enabled more complex KBWSD methods to be devised. The semantic network of WN-like resources can be exploited to derive metrics for the disambiguation of words in context that go beyond naive methods like the calculation of definitions overlap. *Semantic similarity measures* based on structured resources are somewhat akin to the Lesk algorithm, but with them closeness of meaning is calculated on the basis of semantic representations. A target word is disambiguated by selecting that sense which yields the greatest score $\hat{S}$ (Navigli, 2009):

$$\hat{S} = \underset{S \in Senses_D(w_i)}{argmax} \sum_{w_j \in T : w_j \neq w_i} \max_{S\prime \in Senses_D(w_j)} score(S, S\prime)$$

[9]For more information on word embeddings, see section 3.3

where T is the text $(w_1, w_2, ..., w_n)$. Thus, evaluating the *score* function becomes the central issue under this approach (note that this formulation also suffers from exponentially rising complexity when senses need to be selected for all target words in a context, in a globally optimal way). Different similarity measures have been put forward. The most basic one using WN calculates the number of graph edges standing on the shortest path between the two senses (Rada et al. (1989); by "graph" here is meant the WN hierarchy, i.e. the hypernymy/kind-of relations). More complicated measures take into account the number of vertical turns taken in traversing the WN hierarchy (i.e. do you only go from more to less specific or vice versa, or do you go both up and down the hierarchy of kinds; the more times you do it, the more tenuous the link is) and the relative depths of the two senses (Sussna, 1993), the subgraph density of their common ancestor (Agirre & Rigau, 1996), etc. Semantic similarity approaches do not seem to exploit a rich enough theoretical representation of lexical meaning, as they do not really achieve higher accuracy compared to the Lesk algorithm (Navigli, 2009).

More sophisticated methods that rely on lexical representation have been proposed. Analyzing lexical chains (Halliday & Hasan, 1976) in text as mechanisms for maintaining discourse cohesion is the basis for many of those. Lexical chains consist of a series of lexically related words that is woven through the text, bridging sentences and paragraphs. Since a meaningful text is likely to contain different lexical chains, algorithms explore that property in determining the most probable senses of words – the senses selected are part of potential chains with greater weights (where the weight is determined by the types of relations found in the chains and the distance in text over which they obtain; see Galley & McKeown (2003)). Navigli & Velardi (2005) present an algorithm for KBWSD based on lexical chains that first identifies monosemous words and assigns the relevant senses to them, then iteratively selects word senses for the polysemous words that are most strongly connected to the already disambiguated senses via relations from the KB. A number of hand-written semantic pattern rules are used to constrain the calculation of similarity based on the lexical chains.

Another popular alternative are the so called *graph-based methods*. Under them, a ranking algorithm is typically used that calculates the relative importance of the nodes in the semantic graph. Since the nodes are actually the word senses, the most highly ranked ones are returned as disambiguation choices. Mihalcea (2005) constructs a weighted graph with edges between all possible word senses in the current context, where the weights are calculated using the Lesk algorithm. Then

a random walk algorithm (PageRank; Brin & Page (1998)) is used to iteratively calculate the importance of nodes from a global perspective. This overcomes the combinatorial explosion of having to calculate the similarity of all pairs of possible senses, one of the major setbacks in similarity-based approaches. Navigli & Lapata (2007, 2010) describe a two-stage process for graph-based WSD. First the whole KB is explored in order to find a subgraph that is most relevant to the current context. Then a graph-based centrality algorithm is used to find the most important word senses in the subgraph. PageRank again gives some of the best results.

Agirre & Soroa (2009); Agirre et al. (2014) take a somewhat different approach in that they do not make the ranking calculation over a subgraph of the KB. Instead, they use the whole graph, while the probability of initiating a new random walk is done in two ways: through the standard, *static* interpretation of restarting the random walk algorithm and through its *personalized* interpretation, whereby certain nodes in the graph are given greater weight (and therefore their immediate surroundings are as well). The formula for calculating the PageRank vector $\mathbf{P}$ (the importance of all nodes in the graph) is the following:

$$\mathbf{P} = cM\mathbf{P} + (1 - c)\mathbf{v}$$

where M is an N x N transition probability matrix (N being the number of nodes in the graph), c is the damping factor (usually set to 0.85) and $\mathbf{v}$ is an N x 1 stochastic vector. The second part of the equation gives the probability of randomly jumping to any of the nodes in $\mathbf{v}$ and terminating a random walk. The algorithm runs iteratively until convergence or for a fixed number of iterations. Under the static version, $\mathbf{v}$ is uniformly initiated (with values of 1/N), while the personalized version divides the probability mass among the nodes corresponding to the senses of the target words in the particular context. Personalized PageRank obtains good accuracy scores that, albeit still lower than those of supervised systems, are not too far behind on general domain data and in cases of domain-specific data are able to do even better (Agirre, De Lacalle, et al., 2009). The algorithm, which is implemented in the UKB software toolkit[10], can also be used for word similarity calculation (Agirre, De Lacalle, et al., 2009) and named entity disambiguation (Agirre et al., 2015). Agirre & Soroa (2009) show that extending the semantic graph (in this case adding relations from *eXtended WordNet*) has

[10]http://ixa2.si.ehu.es/ukb/

a positive effect on the accuracy score of the algorithm, i.e. graph density is important. Moro, Raganato, & Navigli (2014) discuss a similar disambiguation system that uses the *Random Walk with Restart* algorithm (Tong et al., 2006) and *BabelNet* (Navigli & Ponzetto, 2012) as a semantic graph.

3.2.3 Applications of WSD

Word sense disambiguation has proved to be one of the most difficult core NLP tasks. Until the rise of statistical machine translation in the 1990s, WSD was deemed a crucial element in MT, since it provides a system with the ability to correctly translate lexical items between languages (i.e. as a – relatively – separate task from that of mapping syntactic structures). However, the difficulty in obtaining high precision and recall has hampered its application in downstream tasks and at present it is not entirely clear that high-accuracy WSD is even necessary. The uncertainty has had an impact on the quantity of available data and empirical work, making it harder to achieve satisfactory levels of accuracy, which in turns feeds back in a vicious circle into the initial concern (i.e., confirming the necessity of performing WSD). This difficulty is due to at least two aspects of the problem that need to be balanced. On the one hand, the learning task itself is very challenging and requires rich and smartly handled representations, as well as good algorithms. On the other, there is the issue of how word senses partition the lexicon. Too granular word senses are too difficult to label correctly, while the too coarse-grained do not give much meaningful information. In addition to that, it is perhaps the case that different NLP applications can best make use of word sense representations at different levels of granularity. Nevertheless, a brief discussion is offered below, on the various potential applications of WSD, with a reference to some work done in parallel to the research that constitutes the main focus of the thesis.

Navigli (2009) focuses on several potential such areas. Information Retrieval (IR), the author argues, could benefit from WSD if content words both in queries and in potential document results are disambiguated. In that way, only the pertinent meanings of the search terms will be taken into consideration and search results will be more adequate to the needs of the user. Another potential advantage is the ability to link synonymous word senses in documents, or senses that are related in a relevant way to the search terms. Sources cited by the survey give conflicting information regarding the actual usefulness of WSD in IR. For

instance, Sanderson (1994) argues that only WSD with accuracy over 90% can contribute positively to IR. However, Stokoe et al. (2003) shows that even at a level of accuracy of around 62% WSD can still lead to small gains in IR. Testing the 90% hypothesis is not easy, as sense-annotated data is not easily obtainable in the necessary quantities; Schütze & Pedersen (1995) show that such accuracy in WSD does indeed lead to a significant boost in IR systems.

Another field whose tasks involve locating specific kinds of information – Informaton Extraction (IE) – also should benefit from accurate WSD. Named entity recognition (NER), for instance, has been explicitly and implicitly reframed as a disambiguation problem (e.g. Agirre et al. (2015), as already mentioned, or Ciaramita & Altun (2006), who have implemented a tagger for the 41 supersenses for nouns and verbs in WN and thus obtain information about entities belonging to traditional NER classes like person, location, organization, etc.). In task 8 of SemEval-2010 (Hendrickx et al., 2009), which deals with another IE task – relation extraction, the best-performing system used rich lexical features, including hypernyms of WN senses, Levin verbal classes, disambiguated results from PropBank and FrameNet parsers, etc. (Rink & Harabagiu, 2010).

WSD can be used as a source of features for models performing other core NLP tasks, such as POS tagging, dependency parsing, semantic role labeling (SRL), etc. For instance, Agirre et al. (2011) show that using semantic class / supersense tags from WN can help improve results in transition-based dependency parsing; MacKinlay et al. (2012) demonstrate that word sense hypernyms can, depending on the system configuration, improve HPSG parse ranking. Brown et al. (2014) recast the task of identifying VerbNet classes for the purpose of SRL as WSD, obtaining significant improvements. Zapirain et al. (2013) in turn investigate the semantic role classification subtask and how selectional preferences can be used to improve results on it (identifying selectional preferences can be seen as a task connected to WSD).

Machine translation is a natural area for the application of WSD. Picking the correct translation of words in context is central to MT and it can easily be recast as a WSD task. However, the influence of WSD research on MT has not been strong, in part, of course, due to the fact that WSD systems have failed to achieve high accuracy scores. Carpuat & Wu (2005) report a negative impact of WSD on BLEU scores. They used a supervised model to provide translation candidates for an SMT system, but this only led to worse results. One hypothesis

for explaining this is that the SMT model is powerful enough on its own and the then-current state-of-the-art WSD cannot provide beneficial information in most cases; another explanation is that WSD systems simply cannot be integrated in a meaningful way with SMT systems. In Cabezas & Resnik (2005), target language lexical items are interpreted as "sense tags", or as soft translation variants for the translation model, which selects the final translation via its language model. The study reported a small improvement against a stronger baseline than the one in Carpuat & Wu (2005). Vickrey et al. (2005) re-conceptualize WSD as translation, whereby the possible senses are target words or phrases learned from parallel corpora; the study does report an improvement in results on word translation and blank-filling.

In Chan et al. (2007) disambiguation is done between alternative translations of source phrases. The selection is done in view of maximizing the length of WSD-proposed chunks of text; WSD scores are also factored in. The approach yields statistically significant BLEU score improvements. In a subsequent study Carpuat & Wu (2007) integrate WSD deeper into SMT. Multi-word phrasal disambiguation is carried out and the resulting improvement is reflected across eight different translation metrics. In this case the supervised WSD system provides additional context, so that correct translations proposed by the baseline SMT system are ranked higher than erroneous ones. In addition, the decoder is able to pick longer translation sequences, which tends to correlate with better translations.

Simov, Osenova, & Popov (2016a)[11], work carried out in conjunction with the main research done for the thesis, utilize the output of a WSD system in order to bolster factor-based SMT in the context of English-Bulgarian and Bulgarian-English translation. The baseline model uses only word forms as factors, while the WSD-inflected models replace the word forms with synset and word translations. The translations are obtained via WSD and a mapping between the Princeton English WordNet and a Bulgarian WordNet (Popov et al., 2014). The word form is thus replaced either with the corresponding synset or with a "representative lemma" for the target language synset (based on frequency counts with regards to the synset lemmas). This approach fails to improve upon the baseline, but it demonstrates that lemma-replacement leads to much bigger improvement than synset-replacement, suggesting that recasting translation as WSD may indeed

[11]Simov, K., Osenova, P., & Popov, A. (2016). Towards Semantic-based Hybrid Machine Translation Between Bulgarian and English. In Proceedings of the 2nd Workshop on Semantics-Driven Machine Translation (SedMT 2016).

be productive, especially if more sophisticated and context-sensitive strategies for lemma-selection are devised. In Simov, Popov, Zlatkov, & Kotuzov (2016)[12], a somewhat similar approach is taken, in that Minimal Recursion Semantics (Copestake et al., 2005) is used for better token-to-token alignment between source and target language text, which is then used for rule-based post-processing of the SMT system output. This approach allows for the transferring via the token-alignments of additional information such as dependency links, word senses and elementary predicates.

3.3 Neural Networks for NLP

This section concentrates on the use of artificial neural networks (NNs) for solving NLP tasks, and in particular the lexical analysis tasks tackled in this work: POS tagging and WSD. Before turning to them, I first direct my attention to neural network language models (NNLMs), since research in that area has yielded tremendous advances in recent years – mostly due to the efficient learning of powerful representations of words and other linguistic units (popularized in the literature as *embeddings*). These representations provide rich input for systems that carry out further analysis of text sequences, as they tend to capture many aspects of lexical knowledge. Such distributional models can be seen as an instance of *feature learning* – the automatic discovery of relevant features for data classification; the generation of embeddings has led to much lesser dependence on *feature engineering* – the laborious process of devising feature sets that can adequately represent input data. The section on NNLMs is mostly a compressed version of Popov (2016b)[13], while the part on WSD with NNs – of Popov (2018)[14].

3.3.1 Artificial Neural Networks for NLP

Artificial neural networks (from now on called simply "neural networks") are named so because they are inspired by biological neurons and the networks that are formed by them. NNs consist of nodes and connections between the nodes.

[12]Simov, K., Popov, A., Zlatkov, L., & Kotuzov, N. (2016). Transfer of Deep Linguistic Knowledge in a Hybrid Machine Translation System. In The Workshop on Deep Language Processing for Quality Machine Translation (DeepLP4QMT) (p. 27-33).

[13]Popov, A. (2016). Neural Network Language Models–an Overview. In The Workshop on Deep Language Processing for Quality Machine Translation (DeepLP4QMT) (p. 20-26).

[14]Popov, A. "Neural Network Models for Word Sense Disambiguation: An Overview." Cybernetics and Information Technologies 18.1 (2018): 139-151.

The nodes, or neurons, can be organized in separate layers. Each neuron can be connected to a number of other neurons, which transmit to it signals of varying strength. A neuron that receives some combined input from its connections carries out some kind of processing (typically a non-linear transformation called "an activation function") and on the basis of that outputs its own activation to further nodes in the network. The connections between layers of neurons are typically represented in matrix format, where each neuron from one layer is connected to neurons from the other layers via a dedicated vector (a row or column in the matrix, depending on the direction). The real numbers that represent these links are called "connection weights" and are dynamically modified as the network learns to carry out specific tasks.

There are two main types of NNs that are of interest to this thesis: *feedforward* and *recurrent* NNs. Before examining how precisely I use those in the NLP tasks explored in the dissertation, those are briefly described here.

Feedforward Neural Networks

Feedforward NNs are the simplest kind of artificial neural networks. They can be composed of any number of layers between and including the obligatory input and output layers – the existence of such intermediary layers makes an NN "deep". *Single layer perceptrons*, for instance, do not have hidden layers and are thus the simplest example of an NN (those types of NNs were famously shown in the past to be unable to learn the *XOR* boolean function (Minsky & Papert, 2017)). Multi-layered feedforward NNs (often called *multi-layer perceptrons*) have additional hidden layers which allow them to learn much more complex dependencies. Learning in feedforward networks can be done in different ways, but is almost always carried out via the *backpropagation* of error gradients – the results obtained at the output layer are compared to a training signal and the divergence between the two values is used via the backpropagation of error gradients in order to optimize the connections between the network layers.

Feedforward networks are also characterized by the requirement that inter-layer connections do not form cycles (Sundermeyer et al., 2015). This prevents them from handling longer contexts, as their context representation capability is restricted to fixing sliding windows centered around particular input elements. This is especially important in relation to NLP tasks, as language is characterized by linearity and thus text/speech is most naturally processed in terms of time

series. Since feedforward NNs can only represent a limited snapshot of the input data, they are not very effective in handling pattens that are encoded over larger time intervals. Such patterns are also known as *long distance dependencies* – a term in linguistics used for syntactic relations between words and phrases that cross the tree structure analysis of a sentence and cannot be analyzed in localized terms; in a more abstract sense the term can be generalized to all kinds of linguistic and extra-linguistic relations that obtain in linguistic series. *Recurrent neural networks* (RNNs) are a more advanced variant of NNs that was developed for the purpose of handling such long time series.

Recurrent Neural Networks

The basic formulation of RNNs is not radically different from that of feedforward NNs (for a good overview of RNNs as used for supervised sequenced labeling, see Graves (2012). Larger contexts are handled in RNNs via looping connections between time states of the hidden layers. This mechanism acts as a kind of dynamic memory for the model and it can now theoretically remember important information from an arbitrary number of steps back. This behavior can be formalized in the following way. As with feedforward networks, for a sequence of inputs x_1, x_2, ..., x_n, the network computes the output vector $\mathrm{y_t}$ via the formula:

$$\mathbf{y_t} = W_{\mathrm{hy}} h_{\mathrm{t}} + b_{\mathrm{y}} \tag{3.1}$$

where W_{hy} is a matrix with the connection weights between two layers (in this case, between the hidden and the output layer), h_t is the state of the hidden layer for the current time step and b_y is the bias for the output layer. The recurrence is introduced with the computation of the values for h_t:

$$\mathbf{h_t} = F_{\mathrm{act}}(W_{\mathrm{xh}} x_{\mathrm{t}} + W_{\mathrm{hh}} h_{\mathrm{t\text{-}1}} + b_{\mathrm{h}}) \tag{3.2}$$

where F_{act} is the activation function in the hidden layer, the first term in the summation is the input times the relevant connection weight, the second one is the vector of the previous hidden state multiplied by the relevant hidden-to-hidden connection weight and the last one is the hidden layer bias.

Thus, the LMs are no longer constrained to learning from mere snapshots of narrowly local data. However, RNNs suffer from the exploding/vanishing gradients

problem. The longer the contexts that need to be handled and (consequently) the deeper the networks handling them, the backpropagated error gradients become much too large or much too close to 0. The former case is easier to solve (e.g. by capping the gradients), but the latter is more difficult to deal with and can prevent an NN from training at all. Because backpropagation works by computing the error gradients for earlier layers via the chain rule, whenever the top gradient is close to zero, the chain multiplication of increasingly small numbers effectively erases the training signal and the NN can barely progress, if that is at all possible.

More sophisticated variants of RNNs have been designed to tackle this problem – *Long Short-Term Memory Cells* (LSTMs) and *Gated Recurrent Units* (GRUs; see Cho et al. (2014)). Even though somewhat simpler than LSTMs, GRUs have been found to perform comparably with LSTMs, in some cases even outperforming them (for example, see Chung et al. (2014); Yin et al. (2017)). In the thesis I use the LSTM architecture in terms of handling recurrences, as it is the more popular variant of the two in NLP and since no great difference between the performance of the two architectures has been demonstrated. Below I provide a brief description of LSTM units, as those will be important in the following chapters.

The *LSTM cell* is internally more complex than vanilla RNN hidden layers. Its cell state is modified by the outputs of several gates: *forget gate*, *input gate* and *output gate*. The LSTM cell has four hidden layers inside, which learn the appropriate weights needed to correctly forget and update (parts of) the cell state. Their outputs are calculated via the following equations (Graves et al. (2013)):

$$\mathbf{i_t} = \sigma(W_{\mathrm{xi}}x_{\mathrm{t}} + W_{\mathrm{hi}}h_{\mathrm{t\text{-}1}} + W_{\mathrm{ci}}c_{\mathrm{t\text{-}1}} + b_{\mathrm{i}}) \tag{3.3}$$

$$\mathbf{f_t} = \sigma(W_{\mathrm{xf}}x_{\mathrm{t}} + W_{\mathrm{hf}}h_{\mathrm{t\text{-}1}} + W_{\mathrm{cf}}c_{\mathrm{t\text{-}1}} + b_{\mathrm{f}}) \tag{3.4}$$

$$\mathbf{c_t} = f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{3.5}$$

$$\mathbf{o_t} = \sigma(W_{\mathrm{xo}}x_{\mathrm{t}} + W_{\mathrm{ho}}h_{\mathrm{t\text{-}1}} + W_{\mathrm{co}}c_{\mathrm{t}} + b_{\mathrm{o}}) \tag{3.6}$$

$$\mathbf{h_t} = o_{\mathrm{t}} tanh(c_{\mathrm{t}}) \tag{3.7}$$

where i, f, c, o correspond to the input gate, forget gate, cell state and output gate activations, σ denotes the sigmoid activation function and $tanh$ – the hyperbolic tangent. The memory learns to selectively decide which pieces of information to keep and which to forget, thus making it possible to remember input from many steps ago and to discard input that is not relevant to the current state.

3.3.2 Neural Network Language Models

NNLMs constitute an important research area in NLP, as they sit in the core of many applications which rely on generating or validating meaningful sequences of human language, on the basis of previously observed context: speech recognition, spelling correction, machine translation, etc. Before NNLMs came into wide use, count-based language models (LMs) represented the state of the art. That is, language modeling relied on counting the co-occurrence of words in large text corpora, in order to estimate the probabilities of sequences of words. In count-based LMs, maximum likelihood estimations are obtained per *n-grams*, i.e. word sequences of length n. This is done by counting the appearances of particular n-grams as opposed to (n-1)-grams containing the same tokens except the n-th one. The apparent advantages of count-based LMs lie in that they are relatively simple to estimate and to interpret, and that the algorithms for decoding the most probable sequences are well-understood (e.g. the *Viterbi* algorithm). The downsides, however, are far from negligible: strong independence assumptions (also known as *Markov assumptions*), data sparseness (any sequences beyond 5-grams are almost never replicated in the corpora), difficulty in handling unknown words.

The above-stated issues are to a great extent addressed by NNLMs. Instead of counting co-occurrences of words, neural networks perform non-linear transformations on input contexts and attempt to predict some unseen part of the text based on those intermediate representations. By updating the connection weights (i.e. the *weight matrices*) between the layers of the NN, the predictions are refined against the available training data, until the model has captured a good generative model of the language. This approach has a number of advantages over count-based LMs. First of all, the transformation of the input (usually *one-hot vectors*, whose size is that of the vocabulary and therefore very big) via the hidden layer connections in effect creates a representation of words that is embedded in a much lower-dimensional space, i.e. the transformed vectors for words and contexts typically have just several hundred positions, as opposed to tens of thousands. These compressed representations result in the clustering of words that are semantically "close" to one another. This largely remedies the handling of previously unknown sequences, as those can now be approximated to other ones that follow similar trajectories through the embedding space (character-based NNLMs have been proposed in order to deal with unknown words – by learning

the embeddings of characters in a similar way and combining them into novel word representations (Ling et al., 2015)). This method has the hidden advantage that the lower-dimensional representations come roughly to encode along their positions certain lexico-semantic categories (such as number, gender, animateness, etc.).

Deep Neural Network Language Models

NNLMs come in several flavors. The earliest attempts at obtaining such representations used feedforward neural networks. These models resemble count-based LMs in that predictions are made on the basis of sliding windows of length *n-1* (i.e. much like n-grams). However, feedforward LMs do alleviate the data sparseness issue and do not make such strong independence assumptions as count-based LMs.

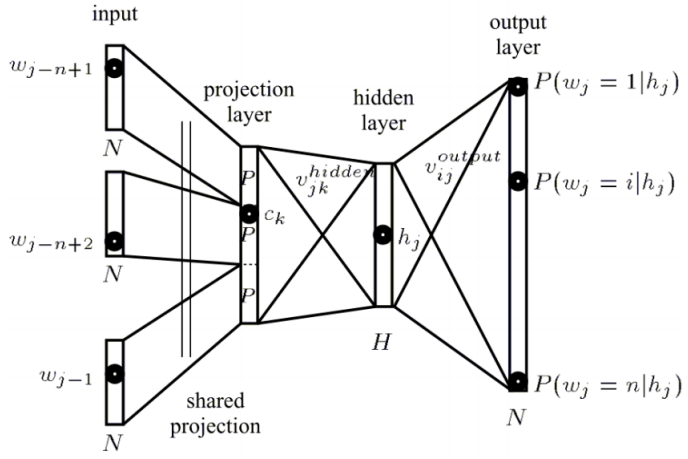

Figure 3.1: Feedforward neural network language model; figure taken from Mikolov et al. (2010).

Figure 3.1 represents graphically a feedforward LM. The sliding context window selects words which are embedded in the shared space. The vectors for the words are then concatenated and passed forward to a hidden layer, which downsizes the word-context representation to a vector of the size of the embedding space. Finally, a probability distribution function (typicall *softmax*) maps the embedding vector to a vocabulary-sized representation that indicates how much each word in it is activated. This final vector is compared to a gold label where the correct word choice is given all the probability mass (1) and all the rest of the words are kept at 0. The model supplies embeddings for the words in training corpus, all the while their representations get more and more precise with the iterations over the data. Large output vocabularies are sometimes problematic, since they require

heavy computation with large matrices, which can slow down training radically. A possible alleviation of the computational hurdle is using a *hierarchical softmax*: decomposing the final probability into two factors: that of the word belonging to a certain class and the probability of it being a specific word from that same class (Sundermeyer et al., 2015). With a good selection of class inventory, computational complexity can be reduced enough so that training is done in manageable time.

RNNs, particularly LSTM-based architectures, were used to improve upon the performance of feedforward NNLMs. The use of RNN LMs has led to significant reductions in perplexity and word error scores (see for instance Mikolov et al. (2010, 2011). Being able to represent past context and to keep a dynamic memory of observed data is clearly very important for the accurate modeling of language.

Shallow Neural Network Language Models

LSTM networks and other powerful NN architectures have been used in many areas of NLP. However, some of the most significant developments in NNLM research recently are rooted in a much simpler general solution. Mikolov, Chen, et al. (2013) introduced *shallow* NNs as a way of training models on amounts of data that had been until then impossibly huge. These networks do away with the hidden non-linear transformations of deep networks, all that is left between the input and output phases is a projection layer. Thus, training time with these *log-linear* solutions is decreased from weeks to days, possibly hours, on billion-word corpora.

Two architectures were proposed in the cited publication: *CBOW* and *Skip-Gram*. They follow essentially the same principles, but while the *CBOW* architecture makes predictions about a single word characterized by a context of N words, *Skip-Gram* operates in the reverse manner – it tries to guess a surrounding context of N words on the basis of the word in the middle. Depending on what training data is available, the two architectures offer different advantages (e.g. typically *Skip-Gram* is more effective with large amounts of data). The removal of the hidden layer, unsurprisingly, harms the precision of the output. The trade-off, however, is worth the sacrifice – this makes it possible to train word embeddings of useful sizes (e.g. 500-position vectors) on data that is big and diverse enough to capture most of the variability found in human languages (e.g. the pre-trained vectors distributed by Google[15] are obtained after training on a 100-billion-words

[15]https://code.google.com/archive/p/word2vec/

corpus). Since then, additional improvements of the models have been put forward, as well as other approaches to embedding itself (see Pennington et al. (2014)) and the embedding of non-word linguistics units (characters, suffixes).

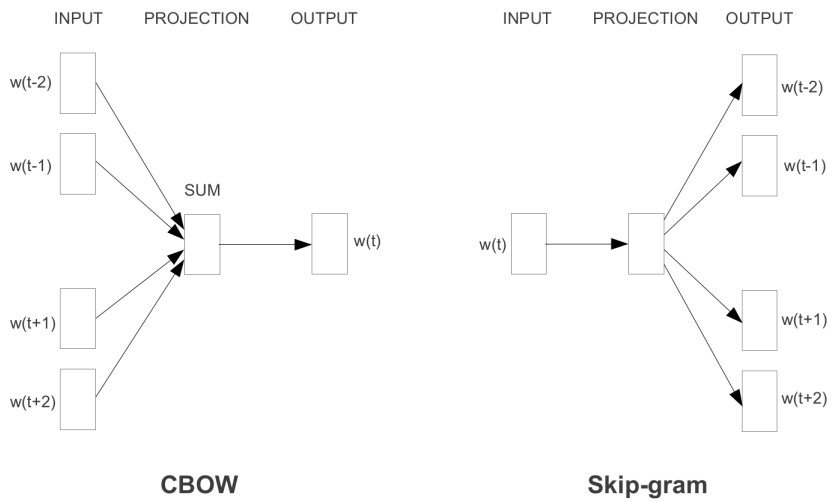

Figure 3.2: The CBOW and Skip-Gram architectures; figure taken from Mikolov, Chen, et al. (2013).

This increased efficiency in learning distributed representations of words has facilitated much of the important NLP research lately. The availability of high-quality embeddings has allowed researchers to move away from the time-consuming process of feature engineering. These developments highlight the power of NNLM to capture dependencies in text. A simple but very effective demonstration of this power is to be found in performing simple algebraic operations on word embeddings. For instance, Mikolov, Chen, et al. (2013) give the following equation:

$$vector(\text{"King"}) - vector(\text{"Man"}) + vector(\text{"Woman"}) \approx vector(\text{"Queen"})$$

Multiple dimensions of meaning are seemingly captured in the same way by word vectors (syntactic, lexical, stylistic, etc.), which suggests that even relatively simple LMs can implicitly capture a great deal of linguistic knowledge that has been modeled before via much more elaborate theoretical constructs.

NNLMs for Representing Concepts, Word Senses and Synsets

The outlined NNLM approaches have been used to learn representations chiefly from natural language data, i.e. word and character embeddings. In theory, however, the same methods can be applied to learning distributed representations

of any kind of unit that occurs in natural series. This idea has been taken up in a line of research that aims to go beyond word representations – to obtain similar information about concepts and word senses. Word embeddings pack a lot of information in themselves, but they bundle together different senses, as many words mask polysemous or homonymous relations between concepts sharing the same form. Different solutions to the problem have been proposed, below I examine a few of those.

The first broad family of approaches is that of *retrofitting methods*, i.e. adapting an already existing resource. Faruqui et al. (2014) is one successful example of this approach, where word embeddings are adapted so as to reflect the structure of a lexical knowledge base. The retrofitting is done through the minimization of an objective function that keeps the new vectors close to the precomputed embeddings (per pair of vectors for the same word) and at the same time keeps words related via the knowledge base close in the embedding space as well. Several sets of word embeddings are retrofitted against three different knowledge bases and improvements are reported across various tasks, with different knowledge bases affecting the word embeddings in specific ways. Johansson & Pina (2015) derive distributed sense representations from precomputed word embedding spaces. They use a lexicon and a semantic network in order to describe words as convex combinations of vectors corresponding to their senses. Sense vectors are subject to an additional constraint – that they remain close in the vector space to the rest of the senses in their *semantic neighborhood*, which is defined on the basis of the knowledge graph (KG) structure. The thus-defined optimization problem is approximately solved with respect to the squared Euclidean distance between sense vectors and their neighbors. The senses and their neighborhoods are based on SALDO, the Swedish semantic network (Borin et al., 2013), and the evaluation is carried out with respect to the task of mapping senses to FrameNet semantic classes. Using the same SVM classifier with sense embeddings as features gives significantly better results than the analogous experiment with word embeddings.

Rothe & Schütze (2015) is another contribution to the line of work which seeks to extend existing vector space models (VSMs) for words to other linguistic units, such as synsets and lexemes (synsets are defined there as sets of synonyms, while lexemes are pairings of words and synsets, or put in another way, lexemes correspond to word senses). The system implemented – AutoExtend[16] – also uses a lexical resource to constrain existing word embeddings and to extend them to

[16] `http://www.cis.lmu.de/~sascha/AutoExtend/`

the new unit representations within the same space. AutoExtend interprets words as the sums of their attendant lexemes, and synsets as the sums of theirs in turn. This maps the three different units in the shared space. The learning of the new embeddings is done with autoencoders, where the input and output layers are fed with the word embeddings, while the hidden layer computes the vectors for the synsets. The transition from word to synset embeddings determines the lexeme embeddings. Constraints derived from WordNet relations keep similar synsets close together in the embedding space.

Camacho-Collados et al. (2015) put forward NASARI[17] – a system for automatically constructing vector representations for WordNet synsets and Wikipedia pages (each page being thought of as a concept in its own right). The BabelNet mapping between WN and Wikipedia concepts is exploited in order to derive two vectors per concept: a word-representation vector based on the Wikipedia page and other pages linked to it, and a synset-representation vector based on the WN synsets for the words in the set of concept-related Wikipedia pages. A technique for measuring lexical specificity is used for calculating the vector weights and synsets are also clustered using the WN hierachy, so that the synset vectors' dimensionality is significantly reduced. The two vector representations per concept can then be used for different applications; the paper evaluates the vectors on word similarity measurement and sense clustering, obtaining very competitive results in all cases. This work does not rely on NNLMs, but rather on the structure and correspondences between lexical/knowledge resources.

Another way to learn distributed representations of senses is to automatically sense-annotate huge amounts of natural text and then train an NNLM on the processed data. Iacobacci et al. (2015) use the BabelNet sense inventory and the Babelfy KBWSD tool (Moro, Cecconi, & Navigli, 2014) to tag a dump of Wikipedia. Only senses chosen by the tool with confidence above a specified threshold are selected at the end. This produces sense annotations with high confidence scores for more than half the open class words: about one billion tagged words and 2.5 million unique word senses. The annotated corpus is used to train an NNLM and the resulting sense embeddings are shown to perform at the state of the art on similarity and relatedness datasets, i.e. working with specific senses is, as expected, preferable to working with word forms. However, this approach has several disadvantages: 1) WSD systems suffer from low accuracy when using granular lexicons (such as WordNet) and therefore this introduces a significant

[17] `http://lcl.uniroma1.it/nasari/`

amount of noise in the training data for the NNLM; 2) even with large corpora, it is difficult to get full coverage of the word senses in a lexicon; 3) it produces embeddings for senses only (since words are replaced with senses in the text).

Mancini et al. (2016) propose an interesting solution – called *SW2V*[18] – to some of the identified problems. Their solution works at several steps. First, they use a *shallow word-sense connectivity algorithm* in order to annotate the open class words of a large corpus with potential word senses. The algorithm relies on the interconnections (i.e. semantic relations from a KG, in this case BabelNet 3.0) between the potential word senses in a given text (in reality this is a sentence or a paragraph). Senses that have a connectivity measure above a certain, optimized threshold are connected to the word forms used in the text. The procedure is linear in time and has the added advantage that it often connects more than one sense to a word, thus alleviating the granularity problem of lexicons mentioned earlier. The next component of the process is a reformulation of the CBOW architecture (Mikolov, Chen, et al., 2013), the difference being that it allows senses (or synsets, supersenses, images, etc.) to be added to the input context and to the output labels. Different configurations are thus possible (words and senses as input & words and senses as output; synsets as input & words and synsets as output, which is the best-performing one; etc.). The SW2V-trained embeddings are shown to perform very well on several tasks: word similarity, sense clustering, induction of *most common senses*.

Chen et al. (2014) is an earlier effort that combines some features of the retrofitting approach, training on a sense-tagged corpus and adapting an NNLM for learning sense embeddings together with word embeddings. First, word embeddings are learned and then those are used to initialize sense embeddings by combining the word vectors for relevant open class words in the sense glosses in WN. These are used to execute a simple WSD algorithm based on comparing potential sense vectors with context vectors for the relevant sentences. The sense annotations that are above a certain confidence threshold are used in a next round of training with a modified Skip-Gram architecture, where context words and senses are predicted in tandem. This method achieves state-of-the-art or competitive results on word similarity, domain-specific WSD and coarse-grained WSD.

[18] `http://lcl.uniroma1.it/sw2v`

A somewhat different way of creating distributed representations of concepts is via generating artificial sequences on which NNLMs can be trained. Goikoetxea et al. (2015), for instance, use a random walk algorithm to traverse a knowledge graph (in their case, the KG is WordNet). The walks can be restarted at each step with some probability and each one of them eventually generates an artificial "sentence". That is, the relational structure of the KG provides the connections between concepts/synsets and the artificial sentences come to reflect this type of conceptual information. The sentences may consist of the node identifiers, or in the case of WordNet as KG they can be replaced by a lemma chosen from the particular synset; hybrid sentences can be produced as well, if the method is configured to replace synset IDs with lemmas with a certain probability. Once an artificial corpus is created in this way[19], the word2vec tool[20] is used to obtain a distributed representation. The authors train *lemma embeddings* and evaluate them on popular datasets for word similarity and relatedness. They report very competitive results compared to the word embeddings learned via the Skip-Gram architecture in Mikolov, Chen, et al. (2013), as well as results comparable to the state of the art when using a combination of resources (such as random walk embeddings + Skip-Gram on text).

Goikoetxea et al. (2016) explore different strategies for producing such combinations and arrive at the conclusion that embeddings learned from separate data resources tend to be complementary and even simple strategies like vector concatenation yield very good results (sometimes better than more sophisticated combination methods). Ristoski & Paulheim (2016) study a similar approach to learning embeddings from an RDF database – they use both random walks and a kernel-based method to generate their artificial corpora. This family of approaches offers two big advantages: 1) no annotated data is necessary for training the models; 2) the representation is built from knowledge encoded in the KG, which is often complementary to what can be extracted from natural texts. This approach to a great extent sidesteps many of the issues discussed above (it has wide coverage, puts the word and sense vectors in the same space, there is no noise from inaccurate WSD), but in turn is not able to explore data from natural texts.

[19]This work uses the UKB tool and its function for printing random walks in order to generate the corpus. I also make use of this functionality: `http://ixa2.si.ehu.es/ukb/`

[20]`https://code.google.com/archive/p/word2vec/`

3.3.3 Neural Networks for Sequence Labeling

In the following section I examine the application of NNs to two specific sequence-to-sequence NLP tasks, i.e. problems where a series of inputs must be transformed into an output series of elements drawn from a different pool of classes. The tasks which the thesis focuses on are POS tagging and WSD. NNs, and more specifically RNNs, are selected as a primary supervised method because they provide a convenient and powerful approach to capturing variable-length contexts. Many applications built to solve these two tasks, especially more recent ones, utilize the previously described LSTM architecture.

There is one simple modification to LSTM networks that is very popular for sequence-to-sequence tasks in general: making them *bidirectional* (Bi-LSTM; Schuster & Paliwal (1997)). This merely stands for the fact that the input sequence is fed twice to two different LSTM layers – the first one is given the input as is ($\{w_1, w_2, ..., w_n\}$), while the second one receives it in reversed order ($\{w_n, w_{n-1}, ..., w_1\}$). The two LSTMs produce a sequence of outputs: $\{h_1^{forward}, h_2^{forward}, ..., h_n^{forward}\}$ and $\{h_1^{backward}, h_2^{backward}, ..., h_n^{backward}\}$. The separate representations from the hidden layers are combined per input step – usually through concatenation ($h_i^{forward} \circ h_i^{backward}$), but non-linear combinations can be applied as well (e.g. $\tanh(L^f h_i^f + L^b h_i^b + b_l)$, where L^f, L^b and b_l are parameters that specify how the combination is done (Ling et al., 2015)) – and then fed to a classifier over the set of possible tags (e.g. a softmax layer). This modification of the recurrent architecture allows the model to look forward *and* backward from the word/token that it tries to tag/disambiguate, an ability that is very important in sequence-to-sequence tasks in NLP, as in language important information is often withheld until a future moment.

Neural Networks for POS Tagging

Collobert & Weston (2008) is a slightly earlier but very influential work on POS tagging with neural networks. In it, a *convolutional neural network* solves in parallel several NLP tasks (such as POS tagging, chunking, named entity recognition, semantic role labeling). Instead of sequential processing, convolutional networks examine a snapshot of the whole context and learn to discover patterns in it. In this way information is shared between the different detection and

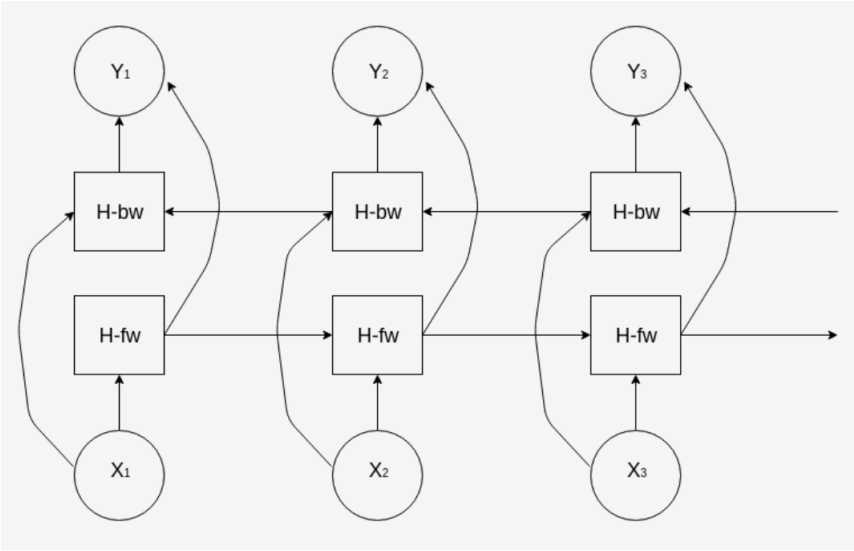

Figure 3.3: A bidirectional RNN (Popov, 2016b)

classification tasks, which helps better constrain all of them and is also an example of *multi-task learning*.

Wang et al. (2015a) is a more recent work, in which POS tagging is treated as a sequence-to-sequence task. It employs a bidirectional LSTM network and achieves results competitive with the state of the art. It also examines different strategies for encoding input data: from one-hot vectors combined with binary encodings for case marking to specially-trained word embeddings, whereby the embedding procedure doesn't learn to predict withheld words but rather to guess whether it is presented with a "correct" context (i.e. a naturally occurring sequence) or with an "incorrect" one (where a certain word in the natural sequence has been swapped with a randomly chosen alternative). Through the addition of a morphological feature (bigram suffix encoded as a one-hot vector) the network is able to achieve state-of-the-art results. Wang et al. (2015b) show how the same architecture can be successfully used for solving other sequence-to-sequence tagging tasks: chunking and named entity recognition. Huang et al. (2015) combine a bidirectional LSTM network with a CRF layer for POS tagging, chunking and NER. They note that this architecture is less dependent on embedded input, achieving accurate scores without recourse to word embeddings.

Plank et al. (2016) evaluate a bidirectional LSTM POS tagger on data for 22 languages and obtain state-of-the-art or close to state-of-the-art results on all of them. Character-based embeddings are shown to be especially useful when tagging non-Indoeuropean and Slavic languages. Another innovation is an auxiliary loss

function that forces the network to distinguish between rare and common words (another instance of multi-task learning; the logic behind this addition is that in this way the network will learn when to trust features pertaining to more frequent words). Ling et al. (2015) is another work that effectively demonstrates the benefit of character-based word embeddings – that a Bi-LSTM model can successfully learn to compose words out of smaller units (characters) and to incorporate both lexico-semantic and lexico-syntactic information in the composition. Such models handle better morphologically complex languages (e.g. Turkish) and have the added advantage of being able to give meaningful representations to out-of-vocabulary words.

With regards to POS tagging in Bulgarian, Simov & Osenova (2001) describe a hybrid approach: a simple RNN combined with a rule-based system. This system achieves "95.17% accuracy for POS disambiguation and 92.87% for all grammatical features". Simova et al. (2014)[21] report on the accuracy of several POS taggers against the BulTreeBank data: 95.91% (BLL Tagger), 94.92% (Mate morphology tagger), and 93.12% (TreeTagger). This work uses the full BulTreeBank tag set of 680 labels. It is also interesting in that it exploits the interaction between multiple dependency parsers and POS taggers in order to improve accuracy scores and can thus be seen as a roundabout instance of multi-task learning.

Neural Networks for WSD

There are several strategies for using RNNs for WSD. One is to use such an architecture in an identical way to what was described with regards to POS tagging – the RNN outputs a hidden layer representation per each input, which is subsequently what the classifier trains upon (be that a softmax activation function, a CRF layer, etc.). Kågebäck & Salomonsson (2016) train a Bi-LSTM model to solve the *lexical sample tasks* of Senseval-2 (Kilgarriff, 2001) and Senseval-3 (Mihalcea et al., 2004), i.e. the network only has to deal with one disambiguation case per sentence. The Bi-LSTM takes both left and right contexts surrounding the target word, reformulated as the corresponding states of the forward and backward LSTMs, and then reshapes that representation through a lemma-specific hidden layer. Thus, there is a separate model trained per lemma that reshapes

[21] Simova, I., Vasilev, D., Popov, A., Simov, K., & Osenova, P. (2014). Joint Ensemble Model for POS Tagging and Dependency Parsing. In Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-canonical Languages (pp. 15-25).

the output of the Bi-LSTM into a vector whose dimensions match the number of word senses of the lemma. The Bi-LSTM layer, however, is shared among the lemma models and continues to learn throughout all training cases. The presented architecture achieves state-of-the-art results and has the added advantage of not requiring any feature-engineering apart from word embeddings as input. However, a significant disadvantage is that separate *lemma experts* have to be learned for all words in the training data – much like SVM-based systems – and thus the model is much less flexible in terms of both training and application.

Raganato, Bovi, & Navigli (2017) propose several neural architectures in order to overcome this problem. They present a Bi-LSTM tagger that learns to disambiguate an input sequence of words $\{w_1, w_2, ..., w_n\}$ drawn from vocabulary V into a sequence of words and senses $\{y_1, y_2, ..., y_n\}$ drawn from vocabulary $O = S \cup V$, where S is the pool of known senses. Two versions of the tagger are put forward – a vanilla Bi-LSTM one and another that adds an *attention layer*, i.e. a mechanism that learns to pay attention to different parts of the context depending on the current situation (Bahdanau et al., 2014). The attention mechanism leads to good improvements on a number of evaluation datasets. Another model is proposed, this time an encoder-decoder architecture, whereby the input sequence is first read by a Bi-LSTM encoder and then a Bi-LSTM decoder takes the encoded representation and "translates" it to the output sequence (Sutskever et al., 2014). The encoder-decoder alternatives do somewhat worse than the Bi-LSTM taggers, indicating that this type of architecture, even though computationally more complex, is less optimal for WSD. This work also introduces two auxiliary loss functions, i.e. a multi-task learning setting – POS tagging and coarse-grained semantic tagging (or supersense tagging). Of the two, only the latter one is shown to benefit WSD. Overall, the attentive Bi-LSTM variants are able to attain state-of-the-art or comparable results.

The other popular and successful strategy for neural WSD, very broadly speaking, aims to represent the context of the target word as a vector in an embedding space and then compare that representation to precomputed embeddings for the set of possible senses that apply to the lemma of the target word. LSTM models offer a powerful tool for representing context in the above-stated way, due to their ability to handle variable-length sequences and to learn both local and long distance dependencies. Such context representations are useful in a variety of NLP tasks, not merely in WSD. For instance, Kiros et al. (2015) describe a model that learns a generic sentence encoder that provides distributed representations

which can be used in different settings. The architecture consists of an encoder that encodes the input sentence and of two decoders which try to reproduce the preceding and following sentences respectively. Melamud et al. (2016) present another neural model for context representation learning, in which a Bi-LSTM layer obtains representations of the left and right-adjacent contexts of the target word; their concatenation is passed to a multi-layer perceptron and the resultant context embedding is used, together with precomputed word embeddings, to obtain the *negative sampling* loss from Mikolov, Sutskever, et al. (2013), on which the model is trained. It is evaluated on several tasks, including WSD, and the system – called *context2vec* – is actually the second-best one in the comprehensive evaluation by Raganato, Camacho-Collados, & Navigli (2017). Other neural approaches to learning context representation are: Kenter et al. (2016), which learns word embeddings optimized for context representation through simple linear combination; Socher et al. (2011), which combines the word vectors in the context according to the structure of its parse tree; Q. Le & Mikolov (2014), which learns to embed words and paragraphs in parallel. Systems of this kind usually also compute sense representations on the basis of example sentences found in dictionary definitions, so that they can calculate a distance measure between those and the context representation.

Yuan et al. (2016) take a similar approach, in that they first train a large LSTM model to encode contexts with respect to a hidden target word and then use that model to compute the representations of example sentences per word sense. New cases for disambiguation are situated in the embedding space and categorized according to their proximity to known instances. The study finds that extending the number of available examples per word and propagating sense labels to them from the gold data (again, via a similarity metric computed over the context representations) leads to improvements in the accuracy scores. In fact, this work reports the highest results in WSD on several popular evaluation datasets. M. Le et al. (2017) note that the above study is difficult to replicate, since neither the data (a large 100-billion-words corpus) nor the code are made available. Theirs is a replication study that tries to reimplement the same system and train it, albeit on a much smaller corpus (1.8 billion words). Despite this limitation and not being able as of yet to replicate all steps of the algorithm in Yuan et al. (2016), they show that their system achieves competitive results. However, even with this much reduced amount of data, training can easily take

months on a powerful machine, so at this point it remains dubious as to how easy it would be for researchers to retrain this system.

3.3.4 Multi-task Learning with Neural Networks

On several occasions in this chapter I have mentioned in passing instances of *multi-task learning*, mostly in relation to NN learning setups (Collobert & Weston, 2008; Plank et al., 2016; Raganato, Bovi, & Navigli, 2017). By this is meant the combination of two or more training signals, against which an NN optimizes its parameters. Different tasks can be combined, e.g. POS tagging and syntactic parsing, or syntactic parsing and token frequency estimation in the context of NLP. Typically, the part of the NN that attempts to answer a particular sub-task is implemented as a separate output layer (or a number of layers) on top of which classification can be carried out. The separate sub-task computational paths in the NN graph may diverge at the same place (e.g. after an LSTM layer) or the task-specific layers may be connected at different places (e.g. having a POS tagging layer after the fist LSTM layer, then having the syntactic parser pathway diverge after the second LSTM layer, etc.); the two training signals can be fed simultaneously at each step, or they may take turns. The network thus shares a number of parameters in the hidden layers that are on the main computational pathway and updates them separately in relation to the training signals.

The main motivation for multi-task learning stems from the hypothesis that different kinds of analyses rely on different kinds of structures in language. Therefore utilizing more than one training signal can force the network to encode in its hidden layers a representation that reflects the separate types of structures. Since the levels of analysis in natural languages are heavily interdependent, such shared representations may in some cases turn out to be useful across tasks. Thus, the lexical representation of a word may be an indicator of what kinds of syntactic patterns it enters into and vice versa (e.g. knowing whether a particular verb is used in a transitive or intransitive sense is important for carrying out a correct syntactic analysis, just as a correct syntactic analysis can disambiguate the usage of a verb with multiple senses). Usually one of the tasks is conceptualized as the *main* task in the setup and the other(s) is thought of as *auxiliary*, i.e. improvements are expected in the first one, whereas the supporting task(s) serves as a source of stabilization – in order to constrain the noise in the first training signal and to help handle the skewedness of the data.

Alonso & Plank (2017) present a study of different combinations of NLP tasks – primary (frame detection and classification, supersense tagging, NER, etc.) and auxiliary (chunking, dependency label classification, POS tagging, frequency estimation). They conclude that multi-task learning is not always effective and the interactions between the different tasks needs to be considered carefully from an information-theoretic point of view. The present thesis offers some preliminary explorations in combining different NLP tasks that all depend on the lexical representation of words. The motivation for these experiments comes mostly from linguistic theory, but the information-theoretic angle demands a serious treatment in any future work.

3.4 Summary and Motivation

The overview of background and related work in this chapter provides a broad picture of some of the approaches to representing the lexicon of natural languages – for computational purposes. This framework of relevant research is used to position the current work with regards to what has already been accomplished and also to identify areas that are under-researched. Especially important for the dissertation is the overarching goal of bridging sub-fields of work, so that hybrid representations of the lexicon can be obtained and explored in terms of the advantages that they can offer.

The literature survey has introduced WordNet as a chief knowledge resource, as well as its most important characteristics in terms of how it models lexical meaning. As indicated in the chapter, there is a significant number of additional linguistic resources that can be used for the enrichment of the symbolic representations of WordNet. This line of work has been explored by some researchers, but is far from exhausted. The approaches to KBWSD introduced provide a convenient and powerful tool for testing such expanded representations of knowledge from different sources. Furthermore, the chapter has touched upon methods for combining this type of symbolic encoding of lexical knowledge with knowledge derived through statistical methods from naturally occurring data, i.e. obtaining hybrid representations that marry theoretical and data-derived models. The probabilistic representation of linguistic units such as words, senses, synsets, etc. is a fast evolving research area, but as evidenced in the survey, there is still abundant space for the exploration of various methods for data representation. This feeds into

the dissertation's focus on transforming the structure of semantic networks into probabilistic models and on deriving such representations not just of words, but also of word senses and contexts. Such novel representations can have multiple applications. One of them is in the field of WSD, and more specifically in WSD with neural networks. In the thesis I offer two approaches to NN-based WSD, which address some of the issues identified in this chapter: solving the *all-words* WSD task with a single model; enabling an NN to make decisions on all encountered words, not just on such seen in training; obtaining richer input features via accessing different kinds of sources; representing contexts in a richer semantic space; combining aspects of lexical learning in order to improve the robustness of representation. Another application of probabilistic models for lexical representation is as a tool for the analysis of knowledge graphs and the (potentially new) relations that obtain within such graph models. This line of work is also incorporated in the dissertation.

Chapter 4

Recurrent Neural Networks for Part-of-Speech Tagging

This chapter presents a neural network solution to the task of POS tagging, as evaluated on a Bulgarian corpus (Popov, 2016a)[1]. The architecture demonstrates that recurrent neural networks are suitable for sequence-to-sequence tasks in NLP and also serves as a basis for the architectures for WSD developed later on in the thesis. The chapter also describes the process of gathering data for training distributed representations of words and suffixes in Bulgarian. The morphological representations are shown to complement the word representations in a significant way and therefore to encode a significant amount of important information for the modeling of morphosyntactic information in the lexicon.

4.1 Word and Suffix Embeddings for Bulgarian

In order to obtain distributed word representations for Bulgarian, a relatively balanced corpus was gathered, large enough for the training of a shallow NN model. It contains approximately 220 million words. The sources for it include: most of the Bulgarian Wikipedia, a selection of news articles and fiction texts. The total number of unique words is 1.6 million, but only "frequent" ones are considered for the training, i.e. those with at least 5 occurrences in the corpus, which gives a final vocabulary size of about 457,000 words (by words here are meant lowercased word

[1]Popov, A. (2016). Deep Learning Architecture for Part-of-speech Tagging with Word and Suffix Embeddings. In International Conference on Artificial Intelligence: Methodology, Systems, and Applications (pp. 68-77).

forms, as the corpus is not lemmatized). The Skip-Gram Word2Vec architecture was used to train the word embedding vectors. Preliminarily experimentation confirmed the intuition that the relatively small size of the corpus obviates the need for larger vector dimensionality, so the final size of the vectors was set at 200 (e.g. using vectors of size 600 did not result in any gains with respect to the task at hand). Additional manual experimentation was carried out, in which the word embeddings were used to generate answers for the word analogy task (after Mikolov, Chen, et al. (2013)): the system is provided with a pair of words among which obtains some kind of semantic relation and with a single word, to which it is expected to find a partner that stands in the same kind of relation. For instance: the query "Athens is to Greece, as Norway is to—" should result in "Oslo"; "mouse is to mice, as dollar is to—" should yield "dollars"; etc. No such set of word pairs exists for Bulgarian, so the embeddings set has not been comprehensively testes on this task, but impromptu experimentation has shown that the VSM is able to identify correctly many semantic relations and to retrieve the relevant words in the pairs.

Having the word embeddings at place, another procedure for training distributed linguistic representations was devised. This type of representation attempts to capture information about morphological segments, which could be complementary to the syntactico-semantic meaning encoded in the word vectors. A smaller subcorpus of 10 million words was extracted for the purpose (solely from the Bulgarian Wikipedia), which was then preprocessed so that it encodes morphological information only. The words in the running text were replaced with parts of them, here called "suffixes" and meant to roughly correspond to meaningful morphological units occurring at the end of word forms. The suffixes are produced by normalizing the original words, i.e. no actual morphological preprocessing is used. Then the Skip-Gram model was once again used to train on the suffix corpus.

The normalization procedure is done as follows. Words are first converted to lowercase, though information about the original case if preserved through a special encoding. In cases where the word length is more than 3 characters, the last three symbols are kept and the preceding segment is replaced with an underscore character. Tokens with three or fewer characters are kept as they are. Case information is preserved according to the following convention: capital case tokens get the "@" symbol concatenated to their beginning; tokens whose letters are all in uppercase (except one-letter words) get the "$" symbol prepended;

mixed case tokens (i.e. lower and upper case letters) are marked with the "#" symbol. Here are some examples of applying the schema just described:

По = @по
по = по
държава = _ава
Държава = @_ава
АП = $ап
БГНЕС = $_нес
вик = вик
Вик = @вик
ВиК = #вик

The thus-obtained training corpus contains about 20,000 unique frequent suffixes. About 17,000 more are excluded from the vocabulary on the basis of frequency filtering – many of those are probably derived from proper names with non-typical endings, rare words with two or three characters, etc. The reasoning behind this procedure is that as Bulgarian is a morphologically rich language that encodes grammatical information on the word level primarily in its suffixes, these representations can capture information not accessible by merely observing word form co-occurrence patterns. In addition to that the suffix embeddings capture information about text cases and word capitalization, which is often crucial for determining whether a word starts a new sentence, is part of a named entity, is an adjective formed from a proper noun, etc. The dimensionality of the suffix embeddings is set to just 50 positions, to reflect the fact that they are much fewer in number and a smaller corpus should suffice for the training process.

4.2 Deep Learning Architecture for Part-of-speech Tagging

For the reasons already outlined in the subsection on sequence tagging with NNs, POS tagging is here approached via an RNN supervised model. The RNN hidden layers utilize the LSTM mechanism in order to dynamically handle long sequences. The architecture is made bidirectional, so that it can incorporate forward and backward contexts into word representations. The states of the two

LSTM layers are concatenated for each time step, resized through a separate linear transformation and the resulting vector is fed into a softmax classification layer that outputs a probability distribution over the POS tagset. During training this distribution is compared against a one-hot gold label, while in testing mode the highest probability tag is selected as final output.

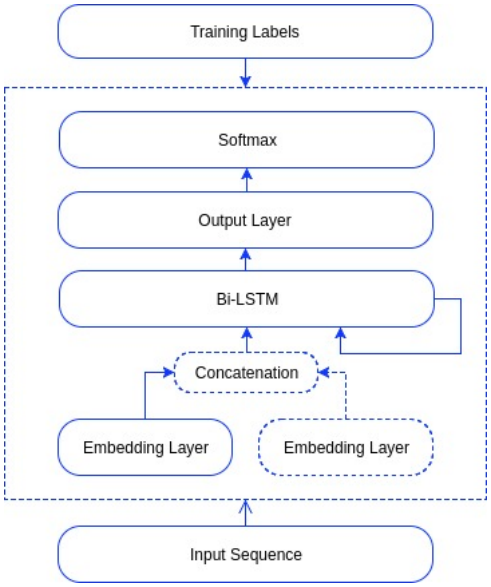

Figure 4.1: Recurrent neural network for sequence-to-sequence tagging: The dotted lines mean that a component or a connection is optional (in the case of concatenating embeddings from two different sources – e.g. word and suffix embeddings). Taken from Popov (2017).

The architecture allows for the concatenation of different inputs – word embeddings and suffix embeddings in this case (see Fig. 4.1). Whenever using larger inputs, it makes sense to use a larger hidden layer in order to handle the increased number of parameters.

4.3 Experiments and Results

The training and evaluation data used in the following experiments come from the BulTreeBank POS tagged corpus (Simov et al., 2002). The corpus contains approximately 38000 sentences, tagged with POS labels. 3500 sentences were used as a validation set, 3500 as a test set and the rest of the sentences were used as training data. The POS tag set used has a medium-level granularity – it comprises of 153 labels. Its labels are more general than those of the full tag set, which number 680, but the tag set, nevertheless, expresses far more morphological

distinctions than tag sets for morphologically poorer languages. For instance, the popular Penn Treebank tag set for English has only 48 labels (Marcus et al., 1993).

A vanilla parametrization of the RNN architecture was used to train the presented models: just one hidden Bi-LSTM layer; learning rate set to 0.3 for all experiments; cross entropy as an objective function; gradient descent as a training algorithm; no regularization techniques. The size of the hidden layer varies slightly in the different experiments. Next, I will briefly describe some of the preliminary experiments and then report the main results of the research.

4.3.1 Setting the Size of the Word Embeddings

Some initial experimentation was done in order to settle on a dimensionality size for the word embeddings used in the main POS tagging experiments. Two sets of embeddings were trained on the corpus – one with vector size 200 and one with size 600. Then the sets were used to provide input representations during the training of a POS classifier. The models trained for only 10000 iterations (per batches of 100 sentences) and had hidden layers of 100 units. Table 4.1 shows the results.

Word Embedding Size	*Accuracy*
200	77.67%
600	71.89%

Table 4.1: POS tagging accuracy depending on the dimensionality of the input word embeddings (after 10000 training iterations)

The results are not sufficient to conclusively prove the superiority of one dimensionality size over the other. For instance, the network could need more time in order to learn all dependencies between the greater number of parameters for the 600-sized set, or it might need a larger hidden layer to do so. Cursory experimentation with a larger hidden layer did not reveal such a pattern, but certainly more empirical data is needed to determine the best parametrization – for the network and the embeddings with regards to the POS tagging task. However, bearing in mind that the training corpus used to learn the embeddings is relatively small (compared for instance to the 100-billion-words corpus used to learn the embeddings in Mikolov, Chen, et al. (2013)), a smaller dimensionality size being a better solution is an intuitively plausible hypothesis.

4.3.2 Suffix Embeddings – Expressive Power and Size

One additional small-scale experiment was carried out to test in advance how viable the suffix embeddings model is. The network architecture was trained for 10000 iterations with only suffix representations per word as input, i.e. no word representations were fed to it.

An additional small experiment is presented here that demonstrates the viability of the approach involving suffix embeddings. The neural network is fed only with vectors from the distributed representations obtained for the suffixes, not making any use at all of the word embeddings. Table 4.2 shows that the suffix embeddings model learns very quickly in the initial stages of training, in fact it performs about as well in comparison to the one with word embeddings only (4.1). This result is not surprising, as morphological units encode more abstract and high-level, grammaticalized information than lexical units. Suffix vectors of two different sizes are used in the experiment: of size 50 and of size 200. The hidden layers are again of size 100 and the experiments are run for 10000 training iterations. The larger embeddings do lead to a slightly better accuracy for this duration of training, but the gap is not large, which more or less matches the intuition that suffix information can be encoded in fewer positions. The smaller vectors were used in the rest of the experiments for the sake of efficiency.

Suffix Embedding Size	*Accuracy*
50	77.11%
200	78.16%

Table 4.2: POS tagging accuracy depending on the dimensionality of the input suffix embeddings (after 10000 training iterations)

4.3.3 POS Tagging with Word and Suffix Embeddings

The main experimental results from this line of research are presented here. Table 4.3 shows the accuracy scores for three different configurations of the neural network. The first model uses only word embeddings as input (of size 200). The second one combines word and suffix embeddings at the input (via concatenation); due to the larger input, it has also a larger hidden layer (125 units per LSTM submodule, compared to 100 in the first model). In order to obtain a fair comparison, a third experiment was done with a model that again uses only word embeddings, but also has a larger hidden layer. All models were trained for 100000 iterations.

Model	*Accuracy*
Word Embeddings (100 neurons)	91.45%
Word + Suffix Embeddings (125 neurons)	**94.47**%
Word Embeddings (125 neurons)	91.13%

Table 4.3: POS tagging accuracy when using word embeddings only, and when complementing them with suffix embeddings (after 100000 training iterations)

The addition of the suffix embeddings clearly increases the power of the POS tagger, thus confirming the hypothesis that morphological information can be captured as efficiently as syntactico-semantic information about words. There is, of course, the possibility that a more fine-tuned configuration of the NN architecture could achieve similar results without using the suffix embeddings (or that the same information from the suffixes could be captured by word embeddings trained on a larger corpus), but much more intensive experimentation is needed in order to test out the various possible combinations. It is also unclear how much useful information in the suffix embeddings comes from the encoding of the word endings and how much from encoding the case of words. However, the results are conclusive enough as a validation step for the usage of RNN architectures as a solution to sequence-to-sequence NLP tasks. The same architecture will be adapted to the more complex task of WSD in a later chapter, and in the final chapter a multi-task model will be presented that attempts to solve the two problems in parallel.

Chapter 5

Graph-based Modeling of Lexical Semantics

The following chapter explores a knowledge-based method for word sense disambiguation that relies on a graph model of the lexicon, as well as strategies for improving its accuracy on the basis of enriching the knowledge graph. The results of several experiments are presented, starting with work tested on a Bulgarian sensed corpus (BulTreeBank), then moving on to experiments with English text as well (the SemCor corpus) and to further elaborations of the strategies for enriching the knowledge graph. The word presented was originally described on the following papers: Simov et al. (2015)[1], Simov, Popov, & Osenova (2016b)[2], Simov, Popov, & Osenova (2016a)[3], Simov, Osenova, & Popov (2016b)[4].

[1]Simov, K., Popov, A., & Osenova, P. (2015). Improving Word Sense Disambiguation with Linguistic Knowledge from a Sense Annotated Treebank. In Proceedings of the International Conference Recent Advances in Natural Language Processing (pp. 596-603).

[2]Simov, K., Popov, A., & Osenova, P. (2016). The Role of the WordNet Relations in the Knowledge-based Word Sense Disambiguation Task. In Proceedings of Eighth Global WordNet Conference (pp. 391-398).

[3]Simov, K., Popov, A., & Osenova, P. (2016). Knowledge Graph Extension for Word Sense Annotation. In Innovative Approaches and Solutions in Advanced Intelligent Systems (pp. 151-166). Springer.

[4]Simov, K., Osenova, P., & Popov, A. (2016). Using Context Information for Knowledge-based Word Sense Disambiguation. In International Conference on Articial Intelligence: Methodology, Systems, and Applications (pp. 130-139).

5.1 Improving KBWSD on Bulgarian Data

As already discussed in the subsection on KBWSD in chapter 3 (background and related work), knowledge-based approaches have enjoyed relative popularity in the research community, due to their limited reliance on annotated data. Typically, however, the knowledge bases used by such algorithms tend to encode mainly lexical and world knowledge: lexical-semantic relations between senses, ontological constraints, etc. Other types of knowledge, more dynamic and determined by experience, are underrepresented in such KBs, even though they contribute crucial bits of information to the representation of meaning in language: collocation of concepts, selectional restriction of words, prototypical arguments of verbs, domain membership, etc. In addition to this, it is an open question whether the existing relations in KBs like WordNet form conceptual networks which are dense enough to capture the paradigmatic knowledge about the world.

This work hypothesizes that such information – additional paradigmatic relations and syntagmatic ones – can be learned from textual resources, as well as from other structured or semi-structured resources that are not explicitly defined in formats such as that of WordNet, or by explicating information contained in WordNet itself. Several such sets of additional knowledge were compiled at the first stage of work and their impact on KBWSD accuracy was tested against a sense-tagged Bulgarian corpus. In the rest of this section I first briefly describe the corpus we used and then the various sets of inferred knowledge and the results obtained with them.

5.1.1 Gold Data and the Inference of New Relations

BulTreeBank is a corpus of texts in Bulgarian annotated with POS and syntactic information (originally in the HPSG framework, a conversion to a dependency format is also available). Sense annotation was carried out after that, including annotation of verb valency frames, senses of verbs, nouns, adjectives and adverbs and entity linking to DBpedia URIs. At the moment when this work was done, part of the sense annotations were mapped to the WordNet 3.0 synset hierarchy and the rest were still in the process of being mapped. The mapping to the Princeton WordNet hierarchy is important because it allows for the reusing of the relations defined over that specific semantic network, whereas the Bulgarian

WordNet still has no associated hierarchy and semantic network defined with the particular language in mind. For a more detailed description of the mapping process, see Popov et al. (2014). Regarding the amount of available data, at that particular point in time about 69,000 word forms in the corpus had mappings to WordNet synsets.

The first set of new relations extracted for KBWSD is not derived from the Bulgarian corpus but rather from WordNet itself. In order to increase the density of the semantic network, the transitive closure of the hypernym hierarchy was computed and made explicit in the conventional format for representing relations in the KB. For example: the WN sense *doctor#1 (doctor%1:18:00::)* is a hyponym of *medical practitioner#1 (medical_practitioner%1:18:00::)* and a hypernym of *surgeon#1 (surgeon%1:18:00::)*, therefore a direct relation is added between *surgeon#1 (surgeon%1:18:00::)* and *medical practitioner#1 (medical_practitioner%1:18:00::)*. The hyponyms of the surgeon sense (and their own) will also be connected to the general sense of medical practitioner, e.g. *cosmetic surgeon#1 (cosmetic_surgeon%1:18:00::)* gets linked to *medical_practitioner%1:18:00::*. This kind of inference is done for all hypernym-hyponym pairs encoded as explicit relations in the WN network and containing synset-IDs found in the Bulgarian WordNet. As a result of this procedure, 590,272 new relations have been added to the semantic network, increasing significantly its density.

The second set of new relations between synsets is syntagmatic in nature. In order to create it, the synactic annotations in BulTreeBank are used in conjuntion with the sense annotations, where such are available. This means in practice that mostly relations between nouns and verbs are extracted, as the rest of the POS categories (adjectives and adverbs) were not yet annotated with word senses at that point (some adjectives did have sense annotations, however). Using the syntactic annotations transformed into dependency format makes the extraction process easy, as words are directly connected via the dependency arcs. This work targets the following dependency relations: *nsubj*, *nmod*, *amod*, *iobj*, *dobj*. The numbers of the relations from these categories that are found in BulTreeBank and both of whose arguments are annotated with WN senses are as follows: 1,844 *nsubj*, 3,875 *nmod*, 1,025 *amod*, 716 *iobj*, and 1,312 *dobj* token occurrences, extracted out of 15,675 dependency relations of these types in total. These relations are transformed to the WN format and added thus to the semantic network.

The relations extracted from BulTreeBank are then generalized to new ones in much the same way as the hypernymy inference is done. The noun nodes in the original relations each provide N hyponyms, which are then inserted in the same relation, in place of their hypernym. This produces N new relations from the original one. Thus, for instance, the relation {*u:00118523-v v:00510189-n*} is derived from an attested nsubj relation, where *00118523-v* is a WN sense annotation of the Bulgarian verb "prodalzha" (continue) and is also the head node in the dependency relation (the predicate in *nsubj*), and 00510189-n, standing for a word sense of "veselba" (revelry), is the dependent node (the subject). The dependent node yields a set of hyponyms which are all (the procedure is recursive, until the whole sub-hierarchy is exhausted) added into a relation with the node *00118523-v*. For instance, *00510723-n* (a synset bringing together particular word senses of the words "binge", "bout" and "tear") has been entered by analogy in the same slot as *00510189-n*.

The outlined procedure relies on a interpretation of the syntactic relations as connecting entities and events (i.e. a general conception of events that includes states as well). If an entity can be related to an event in a particular way, its hyponyms should also be relatable to it in the same way, as they inherit the characteristics of the original entity. Such a strategy suffers from a number of problems: inaccuracies in the original relations (especially when an automatically constructed parsebank is used), non-representative relations (i.e. perhaps the text is very tightly specialized and/or non-standard), the fact that although logically permissible some inferred relations might be highly unlikely. Nevertheless, this approach provides a straightforward means to increase the semantic network density and to add syntagmatic knowledge to the KB. The approximately nine thousand original relations are extended to a new set that contains: 372,247 *nsubj*, 1,125,823 *nmod*, 377,577 *amod*, 114,760 *iobj*, and 292,202 *dobj* relations. Bear in mind that since *nmod* relations involve two entities, they are extended in two steps – one run of the algorithm replaces the head sense with hyponyms and the other does the same for the dependent node.

It is important to make a note that the method outlined above assumes that the relations attested in the treebank do not constitute the most general possible ones. This means that the participants in the events can be generalized upward in the hierarchy. Such an inference mechanism would be inevitably noisy, as it needs to make inductive leaps rather than rely on strict deductive rules. Nevertheless, this work attempted to test naively the utility of such generalization. This further

enrichment is done in the following way. When a particular dependency relation is processed, the entity that will be used to infer new arguments is generalized one level up the hierarchy (i.e. the algorithm climbs up to the hypernym of the selected entity). Then the same procedure as before is applied and all hyponyms of that entity are inserted into the respective slot in the relation. To reuse the earlier example, in the case of the relation {*u:00118523-v v:00510189-n*} the *n*-node will first be generalized to the synset *00509846-n: merrymaking#1 (merrymaking%1:04:00::), conviviality#2 (conviviality%1:04:00::), jollification#1 (jollification%1:04:00::) (a boisterous celebration; a merry festivity).* This synset is then included as a participant in the new relation, and its own hyponym synsets are included as well, i.e. not only the revelry sense but also its other subclass: *00510050-n: jinks#1 (jinks%1:04:00::), high jinks#1 (high_jinks%1:04:00::), hijinks#1 (hijinks%1:04:00::), high jinx#1 (high_jinx%1:04:00::) (noisy and mischievous merrymaking).*

The last set of new relations obtained in the work currently discussed is built on top of the WordNet Domains Hierarchy (Bentivogli et al., 2004). This hierarchy provides domain labels for the WN synsets. Each synset has at least one associated domain, and the domains are specified either by a specific discipline for organizing knowledge (e.g. "linguistics") or by a particular object of interest (e.g. "chess"). First a naive strategy was used under which all synsets related to a domain were connected to a special ID created for the domain. As this did not yield good results, another strategy was employed under which all synsets in a domain were connected to one another. In order to restrict the explosion of new relations, only synsets attested in the Bulgarian data were connected with each other. 132,596 domain relations were obtained in this way.

5.1.2 Experimental Setup and Results

The newly inferred relation sets are used to generate new knowledge graphs to be used with the PageRank algorithm. Since some of the new relations are obtained from the actual gold data, a portion of the BulTreeBank corpus was set aside for testing purposes: out of the 40 files in the corpus, 37 are used to extract the new relations and 3 – purely for evaluation purposes. Those graphs that do not use syntactically-derived relations from the gold data can be tested on the whole of it. Below are given the descriptions and codes for the different combinations of relation sets used to construct the knowledge graphs:

- WN: WordNet relations
- WNG: WordNet relations + relations from the glosses
- WNI: WordNet relations + inferred hypernymy relations
- WNGI: WordNet relations + relations from the glosses + inferred hypernymy relations
- WNGID1: WordNet relations + relations from the glosses + inferred hypernymy relations + domain relations of the kind synset-to-domain and domain hierarchy relations
- WNGID2: WordNet relations + relations from the glosses + inferred hypernymy relations + domain relations of the kind synset-to-synset and domain hierarchy relations
- WNGIS: WordNet relations + relations from the glosses + inferred hypernymy relations + dependency relations from the gold corpus
- WNGISE: WordNet relations + relations from the glosses + inferred hypernymy relations + dependency relations from the gold corpus + extended dependency relations
- WNGISED1: WordNet relations + relations from the glosses + inferred hypernymy relations + dependency relations from the gold corpus + extended dependency relations + domain relations of the kind synset-to-domain and domain hierarchy relations
- WNGISED2: WordNet relations + relations from the glosses + inferred hypernymy relations + dependency relations from the gold corpus + extended dependency relations + domain relations of the kind synset-to-synset and domain hierarchy relations
- WNGISEUD2: WordNet relations + relations from the glosses + inferred hypernymy relations + dependency relations from the gold corpus + extended dependency relations starting from one level up + domain relations of the kind synset-to-synset and domain hierarchy relations

Table 5.1 shows the results for the relation sets that are tested on the full dataset[5]. They suggest, for this particular dataset, that the addition of the transitive closure of the hypernymy relations does improve KBWSD, especially when the new set is added in addition to just the baseline relations from the WN ontology. The domain relations do help only in the case of one-to-one connections between synsets in the same domain.

[5]Page Rank was applied via the UKB tool, which was run with its default settings: a context window of 20 words disambiguated together, after 30 iterations of the algorithm

Combination	*Accuracy*
WN	51.6%
WNG	54.2%
WNI	53.7%
WNGI	54.9%
WNGID1	54.9%
WNGID2	55.1%

Table 5.1: Results on the full gold corpus

Table 5.2 shows the results on the 3 files from BulTreeBank that are not used to extract new relations. This data provide counter-evidence to the hypothesis that hypernymy inference complements the relations from the WN glosses (however, the improvement with respect to the *WN* baseline is consistent with the previous results). The table demonstrates that syntactically-derived relations improve the accuracy significantly. The extended syntactic relations are especially beneficial, giving about a 5% improvement over the original syntactic relations. The more general strategy for extending the relations (going one level up the hierarchy and then down) also accounts for a big improvement (about 3%), indicating that even though this method is certainly quite noisy, the huge increase in relation density in the semantic network offsets this and provides a powerful boost to the KBWSD algorithm. The relations inferred from syntagmatic (i.e. contextual) knowledge account for about a 10% improvement in these results, which means that a dense enough knowledge graph made out of relevant relational information can improve dramatically KBWSD accuracy. Additionally, it indicates that this kind of lexical modeling can capture a lot of information about lexical items. The following subsections provide further investigations of this hypothesis.

Combination	*Accuracy*
WN	51.7%
WNG	53.8%
WNI	53.5%
WNGI	53.7%
WNGID1	53.8%
WNGID2	55.0%
WNGIS	56.5%
WNGISE	61.6%
WNGISED1	61.7%
WNGISED2	62.4%
WNGISEUD2	65.6%

Table 5.2: Results on the test portion of the corpus (3 files)

5.2 KBWSD with Inferred Relations on English Data. Analysis of the WN Relation Types

Next, I provide some further details on the application of the ideas outlined above, only this time tested against English and Bulgarian data. This research is a natural extension of the work presented in the previous section and as such attempts to provide a somewhat deeper analytical insight into this way of modeling the lexicon. First I turn to the work presented in Simov, Popov, & Osenova (2016b) and Simov, Popov, & Osenova (2016a), which analyzes in greater depth the principles and sources of new semantic relations introduced in the previous section. Then I examine the results in Simov, Osenova, & Popov (2016b), which complements further the relation enrichment with two new methods for relation extraction, obtaining additional improvements on WSD accuracy.

Simov, Popov, & Osenova (2016b) and Simov, Popov, & Osenova (2016a) seek to perform a more in-depth analysis of the contributions to the KG quality of various relation sets. To this purpose, the original relations of the WN semantic network are taken per their types and the subgraph of each type set is added to a baseline graph, so that its contribution can be evaluated. A similar method is applied to the relation inference process introduced in the previous section, whereby existing relations are expanded to new ones based on the WN hierarchy; only in this case relation types other than the hypernym-hyponym type are expanded and evaluated. The papers also analyze the separate contributions of the types of relations in the set derived from the WordNet gloss (WNG) corpus and of syntax-based relations.

The baseline KGs used for the evaluation of separate relation types from WN are the following: **WN** (the original relations), **GL** (the relations derived from the gloss corpus) and **WNG** (the combination of the two). In this case the baselines serve not as a lower bound that new accuracies scores should improve upon, but as an upper bound – so that it becomes possible to evaluate the extent to which separate relation sets exhaust the overall contribution of the KG. Both SemCor (English data) and BTB (Bulgarian data) are used for the evaluation (Table 5.3). The same baselines are then used to evaluate the contribution of relations that go beyond the original sets and therefore can overcome them.

KG	*SemCor*	*BTB*
WN	49.37	52.97
GL	51.66	51.15
WNG	58.97	55.90

Table 5.3: Accuracy scores on the two evaluation corpora, when using the original knowledge graphs (Simov, Popov, & Osenova, 2016a).

5.2.1 Evaluating the Original WordNet Relations

For the purposes of the evaluation, the original WN relations are grouped in the following sets, which correspond to the relation types in WN. The nature of the relation is given in parenthesis, the numbers indicate how many such relations are found in the network and the letters in the final parentheses indicate relation subtypes according to what POS combinations are attested in the whole relation set (A – adjective, N – noun, R – adverb, and V – verb).

1. **WN-Hyp** *(hypernymy)* **89089**. (N-N), (V-V).
2. **WN-Ant** *(antonymy)* **8689**. (A-A), (N-N), (R-R), (V-V).
3. **WN-At** *(attribute relation between noun and adjective)* **886**. (N-A), (A-N).
4. **WN-Cls** *(a member of a class)* **9420**. (A-N), (N-N), (R-N), (V-N).
5. **WN-Cs** *(cause)* **192**. (V-V).
6. **WN-Der** *(derivational morphology)* **74644**. (A-N), (N-A), (N-N), (N-V).
7. **WN-Ent** *(entailment)* **408**. (V-V).
8. **WN-Ins** *(instance)* **8576**. (N-N).
9. **WN-Mm** *(member meronym)* **12293**. (N-N).
10. **WN-Mp** *(part meronym)* **9097**. (N-N).
11. **WN-Ms** *(substance meronym)* **797**. (N-N).
12. **WN-Per** *(pertains/derived from)* **8505**. (A-N), (R-A).
13. **WN-Ppl** *(participle of the verb)* **79**. (A-V).
14. **WN-Sa** *(additional information about the first word)* **3269**. (A-A), (V-V).
15. **WN-Sim** *(similar in meaning)* **21386**. (A-A).
16. **WN-Vgp** *(similar in meaning verb synsets)* **1725**. (V-V).

The following experiments provide some indication of the usefulness of these different sets for KBWSD; this is done by measuring the accuracy derived from KGs that employ each of these sets in isolation. These various relation types express different kinds of information and connect different types of nodes in the semantic graph. Here the last property is indicated via the POS combinations, but

in principle other semantic distinctions could be drawn. Therefore, some of the relation types will naturally be useful in isolation only for disambiguating words of certain POS categories. Nevertheless, the relation subset **WN-Hyp**, which concerns only nouns and verbs, is assumed as the most basic relation type and is used in all combinations – because it is much larger than the rest and because it provides a great amount of connectivity, which in a sense makes it a "skeleton" for the KG. Thus, it alone serves as another lower-bound baseline KG and it is always combined with one other relation type in the rest of the KGs in Table 5.4.

KG	*SemCor*	*BTB*	*KG*	*SemCor*	*BTB*
WN-Hyp	33.52	45.03	WN-Hyp+WN-Mm	33.70	44.81
WN-Hyp+WN-Ant	38.63	48.41	WN-Hyp+WN-Mp	35.67	45.22
WN-Hyp+WN-At	36.97	47.91	WN-Hyp+WN-Ms	33.57	45.31
WN-Hyp+WN-Cls	34.23	46.11	WN-Hyp+WN-Per	39.57	48.19
WN-Hyp+WN-Cs	33.54	44.99	WN-Hyp+WN-Ppl	33.53	45.11
WN-Hyp+WN-Der	39.03	50.63	WN-Hyp+WN-Sa	38.29	48.31
WN-Hyp+WN-Ent	33.30	44.65	WN-Hyp+WN-Sim	42.89	49.28
WN-Hyp+WN-Ins	34.18	45.13	WN-Hyp+WN-Vgp	34.22	46.07

Table 5.4: Results for the separate subsets of relations in WN, tested against SemCor and BTB (Simov, Popov, & Osenova, 2016a).

These results demonstrate that not all relation types contribute to the quality of the KGs, as measured for the purpose of KBWSD. There is a number of relation sets that either decrease the accuracy score (WN-Ent) or improve it only negligibly (e.g. WN-Cs, WN-Ms, WN-Ppl). Some of these might be just too small (the WN-Ppl relations number only 79), but this reasoning cannot provide an overall explanation of the differences in the table (e.g. WN-Ms relations are 797 and yield a 0.05% improvement, while WN-At relations are 886 and yet they yield a 3.45% improvement). The greatest improvement is achieved by the WN-Hyp+WN-Sim combination (+9.47%), which is not surprising, as the similarity relation links adjectival synsets – precisely the kind of information that is missing in the hypernymy relations, which do not connect adjectives at all in the KG.

Thus, it is suggested that some of the original relations might in principle be excluded from the KG without hurting the accuracy of the KBWSD algorithm. Some additional experiments have to an extent confirmed this hypothesis. For instance, the combination **WN-Hyp + WN-Ant + WN-Der + WN-Per + WN-Sa + WN-Sim + WN-Mp + WN-Cls** results in an accuracy score of 49.34% on SemCor, only 0.04% lower than the baseline result with the full graph. The same KG performs with only 0.61% lower accuracy on the BTB data. Thus,

almost 25,000 relations are excluded from the KG with relatively small negative effect on the WSD process.

The influence of the various relations sets is also different on WSD as performed on the two corpora. One reason for this might be that the Bulgarian data is much more homogeneous in terms of the domains it covers, in contrast to SemCor; also, BTB is annotated mainly with noun and verb senses.

5.2.2 Inference over WordNet Relations

Similarly, an evaluation is performed of the impact obtained by adding inferred relations to the baseline WN and WNG graphs. The inference is done via the same strategy as was presented in the previous section, but modified per relation type. Thus, WN-Hyp is extended via inferring its transitive closure, but for the WN-Ant type, for instance, only N-N relations are extended, as adjectives and adverbs are not hierarchically organized in WN. The different relation sets require their own inference logic; for instance: **WN-At** (attribute) relations allow for all hyponyms of a noun to inherit its attributes; in the case of **WN-Der** (derivational) relations, which cover a diversity of word-pairs of different POS categories, nouns can be substituted with their hyponyms and verbs can be substituted with their hypernyms; **WN-Mp** (meronymy relation) and **WN-Ppl** (participle relation) are not extended in this work; etc. For more details on the specific ways relations are extended, see Simov, Popov, & Osenova (2016a).

Table 5.5 shows that improvements based on additional relations inferred from the original WN semantic network are minimal, if present. The most significant gains are observed with respect to the BTB data, when the inferred relations are added to the base WN relations – all extensions have a significant positive effect in that case; with SemCor, improvement is registered only with **WN-HypInfer** and **WN-Cs2ndVInfer**, the second being only a negligible one. The inferred relations, together with the WN set, do indeed boost accuracy on BTB markedly, so much that these sets achieve better results compared to the WNG baseline. The combination of WNG + inferred sets does not yield a great improvement. Results on SemCor rise only modestly, if at all, and the same can be said about results on BTB as well. This suggests that perhaps many of the inferred relations provide overlapping information with those from the gloss corpus and therefore they either do not contribute positively or in fact hurt performance.

KG	*SemCor*	*BTB*	*KG*	*SemCor*	*BTB*
WN+WN-HypInfer	**53.40**	**53.70**	WNG+WN-HypInfer	58.59	55.20
WN+WN-AntInfer	48.57	**53.05**	WNG+WN-AntInfer	**59.14**	**55.93**
WN+WN-ClsInfer	48.43	**54.62**	WNG+WN-ClsInfer	57.84	**56.14**
WN+WN-Cs1stVInfer	49.32	**56.02**	WNG+WN-Cs1stVInfer	**59.06**	**55.93**
WN+WN-Cs2ndVInfer	**49.39**	**57.28**	WNG+WN-Cs2ndVInfer	58.95	**56.17**
WN+WN-DerNAInfer	48.76	**57.19**	WNG+WN-DerNAInfer	58.49	52.13
WN+WN-DerNNInfer	47.79	**56.74**	WNG+WN-DerNNInfer	58.80	52.86
WN+WN-DerNVInfer	47.73	**55.84**	WNG+WN-DerNVInfer	55.87	52.77
WN+WN-DerVNInfer	48.72	**55.99**	WNG+WN-DerVNInfer	**59.00**	53.56
WN+WN-Ent1stVInfer	49.34	**56.08**	WNG+WN-Ent1stVInfer	**58.98**	52.55
WN+WN-Ent2ndVInfer	49.36	**56.60**	WNG+WN-Ent2ndVInfer	58.92	52.70
WN+WN-InsInfer	49.03	**56.76**	WNG+WN-InsInfer	58.38	52.89

Table 5.5: Accuracy scores on SemCor and BTB. The left part of the table presents results for KGs combining the original WN relations and one inferred set of relations (denoted by the "Infer"-suffix); the right part presents a combination of the original and gloss relations together with one extended set. Aditional specifiers like "1stV" indicate which part of the original relation is extended (first verb in this case); specifiers like "NN", "NV", "VN" indicate which subsets of the relation set are extended. The results that are higher than the baselines for WN and WNG are bolded (Simov, Popov, & Osenova, 2016a).

5.2.3 Analysis of the Semantic Relations from eXtended WordNet

A similar analysis is offered here of the relations derived from the eXtended WordNet (XWN) corpus (Mihalcea & Moldovan, 2001). Those relations make the **GL** set. They are composed according to the following simple procedure: the synset annotations of the open class words in the WN glosses are connected to the synset corresponding to the particular gloss. This produces a number of relations per gloss (as many as there are annotated open class words in it; relations derived across glosses can be duplicated). For the purposes of this evaluation, the gloss-derived relations are divided into groups depending on the POS category of the synset from whose gloss the relation is constructed. Thus, the **GL** set is decomposed into **GL-A**, **GL-N**, **GL-R** and **GL-V**.

Table 5.6 shows the impact of the separate subsets on the WSD accuracy. All combinations improve the WN baseline against the SemCor data, but with regards to BTB, there is an increase only with **WN+GL-A** and **WN+GL-V**, the first

KG	*SemCor*	*BTB*	*KG*	*SemCor*	*BTB*
WN+GL-A	52.94	53.08	WN+GL-R	51.76	52.85
WN+GL-N	**57.04**	52.92	WN+GL-V	53.01	**56.01**

Table 5.6: Accuracy scores for the combinations of the base WN relations and a POS-determined subset of the GL relations. The highest results for the two corpora are bolded (Simov, Popov, & Osenova, 2016a).

being a very slight one and the second – giving a result that is higher even than the full WNG baseline.

5.2.4 Analysis of the Syntax-derived Relations

As in the previous section, syntactic annotation is used to generate new semantic relations that are then added to the KG. For the purpose of generating the new relations, SemCor was dependency-parsed with the IXA pipeline[6]. 49 of the documents of the corpus annotated with all types of open class words were set aside as a test partition and the rest were used as a source for new relations. These documents served to extract dependency relations between words annotated with synset information; for instance: s_1-*subj*-s_2, where s_1 is a noun synset and s_2 is a verb synset; s_1-*mod*-s_2, where s_1 is an adjective synset and s_2 is a verb synset; etc. Again, the extracted relations are organized into subsets defined by the combination of different POS categories. Thus, the following subsets are formed: **SC-AA**, **SC-AN**, **SC-AV**, **SC-NN**, **SC-NV**, **SC-RA**, **SC-RN**, **SC-RR**, **SC-RV**, **SC-VN**, **SC-VV**.

KG	*SemCor*	*BTB*	*KG*	*SemCor*	*BTB*
WNG+SC-AA	**59.20**	**55.93**	WNG+SC-RN	58.94	55.89
WNG+SC-AN	**59.30**	55.89	WNG+SC-RR	**59.07**	**55.93**
WNG+SC-AV	**59.46**	55.78	WNG+SC-RV	**59.43**	52.71
WNG+SC-NN	58.81	**56.21**	WNG+SC-VN	**59.05**	55.55
WNG+SC-NV	**59.31**	**56.21**	WNG+SC-VV	**59.26**	53.78
WNG+SC-RA	**59.52**	**56.18**			

Table 5.7: Accuracy scores for the combinations of WNG and syntactically-derived relations from SemCor.

Table 5.7 demonstrates that most of the syntactically-derived subsets improve the accuracy scores beyond the WNG baseline, which is consistent with the observations from the previous section on Bulgarian data. The improvement is

[6] `http://ixa.si.ehu.es/Ixa`

more consistent on the SemCor test portion than it is on the BTB data, which is not surprising, considering that syntax-based relations are more language-specific than lexico-semantic relations. Using the syntax-based relations derived from the BTB data (as discussed in the previous section) was tried for the purposes of WSD on SemCor, but this did not lead to any improvements over the WNG baseline, rather it diminishes accuracy. The BTB data is more domain-focused, which, in combination with syntactic difference across the languages, possibly explains the vanishing effect of the inferred relations across corpora.

The combination of the syntactically-derived subsets and the baseline relations that yields the highest results on SemCor is as follows: **WNG**, **SC-AA**, **SC-AN**, **SC-AV**, **SC-NN**, **SC-NV**, **SC-RA**, **SC-RN**, **SC-RR**, **SC-RV**, **SC-VN**, **SC-VV**. This KG results in 60.34% accuracy on SemCor and 53.05% on BTB. This result on SemCor can be further improved by adding some of the inferred relations from the WN semantic network. The best combination – **WNG**, **SC-AA**, **SC-AN**, **SC-AV**, **SC-NV**, **SC-RA**, **SC-RR**, **SC-RV**, **SC-VN**, **SC-VV**, **WN-HypInfer**, **WN-AntInfer**, **WN-DerVNInfer**, **WN-Ent1stVInfer**, **WN-Ent2ndVInfer** – yields 60.70% accuracy on SemCor, which is 1.73% better than the WNG baseline, and 56.39% accuracy on BTB.

5.2.5 Further Explorations of Relation Construction

In this final subsection I describe several additional methods for creating new relations between nodes in the KG and their impact on WSD accuracy. This work is published in Simov, Osenova, & Popov (2016b).

Deriving Knowledge from the Logic Form of Statements

The eXtended WordNet resource, in addition to word sense annotations of the open class words in the WN glosses, provides also parse-trees and logic forms (i.e. first-order logic representation) of the same sentences. If we analyze verbal, adjectival, adverbial and prepositional predicates (i.e. predicates in the logic form that are derived from lemmas of the corresponding POS category), then the arguments of the same predicates are in an "event" relation both with the predicates and among themselves, if more than one argument is present. Separate predicates can also be linked together via a relation, provided they are associated with the same argument in a logic form.

For instance, the logic form of the concept {ice-cream cone}, defined as "ice cream in a crisp conical wafer", can be expressed thus:

```
ice-cream_cone:NN(x1) ->
   ice_cream:NN(x1) in:IN(x1, x2)
   crisp:JJ(x2) conical:JJ(x2)
   wafer:NN(x2)
```

From this formula, it is possible to extract semantic relations between "ice_cream" and "wafer" (via the "in" predicate), between "crisp" and "wafer", "conical" and "wafer", and between "crisp" and "conical". The arguments of the relations are translated via the word sense annotations to WN synset IDs and in this way the logic form is translated to the relation format used for KBWSD. The new graph is called **WNGL**.

Building Knowledge Subgraphs from Context

This approach to creating new relations aims at extracting contextual information about word senses. It utilizes the information in SemCor and eXtended WordNet – sense annotations over natural language that are performed by human experts and are thus highly precise. Two different strategies for creating relation-encoded contexts are explored: making use of word order or of syntactic structure as the "connective tissue" of the context graph.

The first strategy is used to extract contexts from XWN. The extraction procedure is a simple one. All tokens in the gloss that are annotated with synset IDs are integrated in a tree-like structure that represents the context. First, the IDs are converted from the WN 2.0 version to the corresponding WN 3.0 encoding. For each of the tokens that are to be connected in the tree, a special artificial node is created. Then each such node, following the word order of the gloss, is connected via relations to the corresponding synset ID in the annotation of the token and to the artificial node of the preceding annotated token (the first token of the gloss has no connection to a preceding element, it is the root of the tree). Figure 5.1 provides a visualization of the resulting context. This new graph made out of the contexts generated from XWN is called here **WN30glCon**.

The second strategy for context creation extracts relations from the syntactic parses of SemCor sentences. The method is similar to what has already been

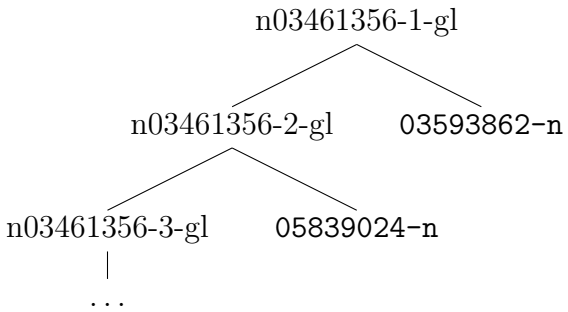

Figure 5.1: A tree structure that represents graphically part of one constructed context. Terminal nodes are represented only by WN synset IDs (Simov, Osenova, & Popov, 2016b).

presented with regards to syntactically-derived relations. However, under this approach synset nodes are not directly connected in the KG based on dependency relations found in the corpus. Rather, artificial nodes are created for the words that bear the synset annotations and these are used to create the parse tree (derived by an automatic parser, therefore at least somewhat noisy). This method preserves information about words that are not annotated with synset IDs by inserting their nodes in the tree; synset IDs, as in the previous strategies, are attached as terminal nodes to the artificial ones; and finally, the top node of each sentence is connected via an arc to the root of the preceding sentence in the current text fragment in SemCor. Thus, each fragment is transformed into a graph which corresponds to one larger context, which itself forms connections between synset nodes within the original WN network. Figure 5.2 provides a graphical view of one part of such a context. The resultant graph is named **GraphRelSC**.

Table 5.8 displays the WSD accuracy scores after running the UKB system with different combinations of the KGs. Two different configurations of the UKB system are used: the *Static* interpretation of the *PageRank* algorithm and its "personalized" variant – *PPRw2w*. The results demonstrate that adding relational information from different knowledge sources can significantly improve KBWSD results. It is also clear that there is significant overlap between the subgraphs – for instance, the last two lines in the table suggest that the **WNG** set of relations does not contain a very significant amount of new information that is not already present in the rest of the sets.

In keeping with the previously discussed findings, figuring out ways to cut down the size of the KG and of simplifying its topology, while preserving information-rich relations, can lead not merely to faster execution time, but also to higher

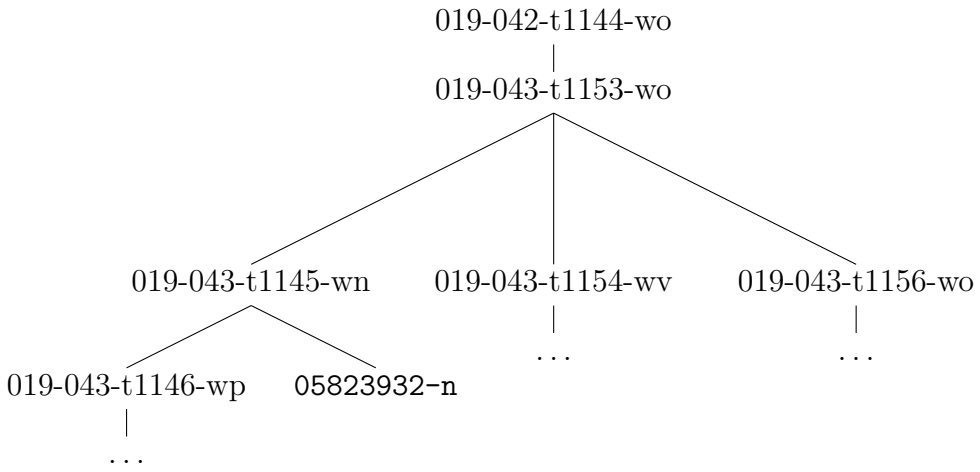

Figure 5.2: The top part of a dependency parse of one sentence, also connected to the previous sentence via a link to a preceding top node (Simov, Osenova, & Popov, 2016b). The numbers in the artificial nodes are simply indices to the file, sentence and token positions within SemCor.

accuracy. Another potential challenge would be to provide additional relations that are able to explicate distinctions between word senses that tend to appear together in common contexts – as this will often mean that they co-occur in the constructed contexts and are thus indistinguishable solely on the basis of this knowledge source.

KG	*Static*	*PPRw2w*
WN	56.60	56.35
WNG	56.00	57.33
WN + WNG	**59.55**	**62.24**
WNGL	60.46	60.35
WN + WNGL	66.61	67.19
WN + WN30glCon	67.00	66.42
WN + GraphRelSC	67.04	65.97
WN + GraphRelSC + WNGL	68.41	68.51
WN + WN30glCon + GraphRelSC	68.74	68.15
WN + WN30glCon + GraphRelSC + WNGL	**68.77**	68.48
WN + WNG + WN30glCon + GraphRelSC + WNGL	68.39	**68.59**

Table 5.8: WSD accuracy scores for different combinations of the already existing and newly created knowledge graphs.

Chapter 6

Distributed Representation of Words, Lemmas and Senses Based on Lexical Resources

In the next chapter I introduce some work on generating distributed representations of lexical units: words, lemmas and senses. The generated vector-space models (VSMs) are of interest because they are trained on data that is artificially constructed rather than extracted from natural texts. This approach, as also discussed in the chapter on the related work, allows for wider coverage and for encoding somewhat different semantic information in the lexical models. The chapter is based on two papers written together with Kiril Simov, Petya Osenova and Iliana Simova: Simov et al. (2017)[1] and Simov et al. (2018)[2]. In the first paper, different knowledge graphs – extensions of the WN semantic network in the spirit of what was presented in the previous chapter – are used to generate artificial corpora on which lemma embedding models are trained and then compared to VSMs trained on natural language text. The second paper outlines an approach to learning distributed representations of grammatical arguments, which are subsequently used as filters to constrain in a meaningful way the already familiar procedures for relation expansion.

[1]Simov, K., Osenova, P., & Popov, A. (2017). Comparison of Word Embeddings from Different Knowledge Graphs. In International Conference on Language, Data and Knowledge (pp. 213-221).

[2]Simov, K., Popov, A., Simova, I., & Osenova, P. (2018). Grammatical Role Embeddings for Enhancements of Relation Density in the Princeton Wordnet. In Proceedings of the 9th Global Wordnet Conference.

6.1 Learning and Evaluating Embeddings from Different Knowledge Graphs

Chapter 3 has already introduced the idea of generating artificial data for the purpose of training embedding models over them (Goikoetxea et al., 2015). The currently presented line of research attempts to build on that previous work and to explore different knowledge graph configurations for obtaining better distributed representations, which can then serve as input features for various NLP tasks, including WSD.

As established in chapter 5, enriching the knowledge graph through the addition of new relations can lead to an increase in the accuracy of the KBWSD algorithm. This is especially true when the new relations are complementary to those already available – in the case of WordNet as KG, syntagmatic relations extracted from contexts of actual linguistic usage have a pronounced effect when added to the largely paradigmatic relations encoded in the semantic network. Syntagmatic relations like co-occurrence of words and syntactic dependencies, on the other hand, are relatively easy to learn using modern methods for learning distributed representations, while the paradigmatic information in KGs like WordNet remains implicit in natural texts. Therefore, one promising line of research is to attempt to improve VSMs as lexical models by combining the two types of knowledge. To do that, we have adopted the strategy presented in Goikoetxea et al. (2015) – of producing artificial sequences of lexical units via the KG and then training VSMs on that data.

The procedure for generating the corpora, training the VSMs and evaluating them is as follows:

1. The knowledge graph is assembled from various sets of relations
2. The UKB tool is used in its "walkandprint" regime in order to produce random walks of variable lengths along the structure of the KG. The tool can be configured to output, with certain probability, synset IDs or the first lemma per synset. When training lemma embeddings, the probability for outputting a lemma is set to 1. Each random walk constitutes one training sentence.
3. The word2vec tool is used to train a VSM on the artificially produced corpus. We have used the Skip-Gram architecture.

4. The resulting VSM is evaluated on the task of calculating relatedness and similarity scores between pairs of words. The Word-353 Similarity, WordSim-353 Relatedness (WS353) and SimLex-999 (SL999) datasets are used for the evaluation (see chapter 2 for more detailed information). The VSM is used to get the embeddings for the word pairs, then those are used to calculate their cosine distance. The distance metrics for all pairs of words are then used to calculate Spearman's rank correlation with respect to the gold corpus numbers provided by human annotators.

In this work, the experimental comparison is done against three VSMs, all of which have a dimensionality of 300:

- The GoogleNews vectors, trained on a 100 billion newswire corpus, using the word2vec tool[3].
- The Wikipedia dependency vectors described in Levy & Goldberg (2014), also trained with word2vec[4]
- The lemma embeddings based on the WN graph and described in Goikoetxea et al. (2015)[5].

The graphs that we have used to produce the new training corpora – in addition to *WN* and *WNG* – are the ones described in chapter 5:

- WNGL – relations extracted from the logic form of the annotated glosses in XWN.
- GraphRelSC – relations constructed from the dependency trees of SemCor sentences analyzed with an automatic syntactic parser. Subsequent sentences are connected to each other via their root nodes.
- WN30glCon – relations built from the XWN glosses, wherein each content word receives an artificial node, which is also connected to the corresponding WN synset ID and to the artificial node of the previous word.
- HypInf – the transitive closure of the hypernymy relations in WN.

We have explored a range of options when configuring the word2vec Skip-Gram architecture. In the experiments the context window around the input

[3]The first two models are downloaded from this url: https://github.com/3Top/word2vec-api. The Google vectors are also available here: https://code.google.com/archive/p/word2vec.

[4]Also available for download here: https://levyomer.wordpress.com/2014/04/25/dependency-based-word-embeddings/.

[5]http://ixa2.si.ehu.es/ukb/

word was varied between 1 and 19 words; the best results were obtained with context windows of 5 and 15 words, which are reported here. Different numbers of training iterations over the corpus were also tried out, the best results were achieved with 7 iterations. The number of noise/negative words was set to 5 and the threshold for the downsampling of frequent words was fixed to 1e-7 after some experimentation.

Vector Space Model	*WordSim353 Similarity*	*WordSim353 Relatedness*	*SimLex999*
GoogleNews$_{\text{Mikolov, Sutskever, et al. (2013)}}$	0.77145	0.61988	0.44196
Dependency$_{\text{Levy \& Goldberg (2014)}}$	0.76699	0.46764	0.44730
WN + WNG$_{\text{Goikoetxea et al. (2015)}}$	0.78670	0.61316	0.52479
WN + WNG + HypInf *C5*	0.77730	0.54419	0.55192
WN + WNG + HypInf *C15*	0.77205	0.55955	**0.55868**
WN + WNglConOne *C5*	0.77761	0.64747	0.53242
WN + WNglConOne *C15*	0.79659	**0.65548**	0.52632
WN + WNG + WNGL + GrRelSC *C5*	0.79847	0.63587	0.51974
WN + WNG + WNGL + GrRelSC *C15*	**0.81862**	0.61455	0.52350
WN + WNglConOne *C15* + **GoogleNews**	**0.82684**	**0.70972**	0.54675
WN + WNglConOne *C15* + **Dependency**	0.80428	0.66570	0.54041

Table 6.1: Comparing results from different VSMs on the similarity and relatedness tasks. *C5* and *C15* are used to indicate the size of the context window for the Skip-Gram model. The best results on the different data sets, using a single VSM as source, are marked in bold. The final lines give the correlation scores for combinations of VSMs: a graph-based one and the GoogleNews/Dependency vectors; the first combination achieves the best overall results on two of the data sets and comes close to the best result on the third one.

Table 6.1 shows the Spearman's rank correlations for the different VSMs, compared with the gold annotations. It is important to note that because the VSMs produced from WN only contain lemma representations and no word form representations, some words in the evaluation data sets have been excluded in order to ensure a fair comparison between the models. 11 pairs out of 203 were excluded from WordSim353 Similarity; 19 pairs out of 252 were excluded from WordSim 353 Relatedness.

Different combinations of subgraphs are shown to be useful with respect to the different data sets. The transitive closure over the hypernymy relations in WN, for instance, leads to a significant increase on the SimLex999 data

set. This could be due to the greater correlation between word similarity and paradigmatic, hierarchical lexical knowledge. Conversely, adding syntagmatic information (WNglConOne) mostly benefits the results on the relatedness dataset, indicating that relatedness is indeed mostly correlated with relations in context. Adding GrRelSC and WNGL to the original subgraphs leads to the best results on the WordSim353 dataset, suggesting that this VSM is based on the densest knowledge graph representation of the lexicon. A final experiment combines two VSMs via concatenation: WN+WNglConOne-C15, which is among the best graph-based models considered, and the GoogleNews and the Dependency vectors. The combinations surpass the correlation scores of all single VSMs that comprise them, and furthermore the first combination achieves the best results on the WordSim data sets, also coming close to the best result on the SimLex data set. This provides evidence that the two sources of information are complementary to some extent and provide access to different aspects of lexical knowledge.

The experimental evaluation shows that the improved performance of KBWSD is mirrored in improved VSMs, when using an enriched KG. In the next section I will discuss one approach to using VSMs generated in this way in order to improve the graph enrichment procedure itself. The next chapter in turn will investigate whether such distributed representations can be useful for improving the performance of supervised methods for WSD.

6.2 Increasing the Density of the Knowledge Graph Through Filtering with Grammatical Role Embeddings

In this section I present an approach to enriching the KG that depends on work described in the first part of the chapter and in the previous one. The idea is to try and increase the density of syntagmatic relations in the KG, which tends to help increase the quality of KBWSD, as already demonstrated (and could serve other, maybe even non-computational purposes). The syntactic relations that can be extracted from a corpus with human-annotated word senses such as SemCor, however, are just a fraction of the possible relations that can be meaningful with respect to world and linguistic knowledge. Performing logical inference over such "gold" relations can be an effective strategy, as shown in Simov et al. (2015) and

Simov, Osenova, & Popov (2016b). In those articles, the dependency relations *subj*, *dobj*, *iobj* and *nmod* that involve two WN-annotated words are extracted and the WN hypernymy relations are used in order to infer analogous relations for the hyponyms of the noun arguments in the "gold" relations.

However, the word sense hierarchy of WN does not guarantee monotonicity in the inheritance of semantic and syntactic features, therefore the outlined method is inherently noisy. For instance, the relation "doctor-operates" extracted from the sentence "A doctor operates on a patient", when extended along the noun's hyponym chain, will result in new relations, some of which false – not all kinds of doctors can operate. In addition to this problem, it has been shown already that the generalization of syntagmatic relations can be broader in scope – this was shown by first generalizing the noun argument to its hypernym and then taking all of the upper node's hyponyms as arguments of new, inferred relations. Such an approach can also improve the quality of WSD (thus indirectly suggesting that the KG has been meaningfully expanded), but the noisiness only becomes greater and there is no good rule that says how high in the hierarchy exactly the method should go in order to achieve maximum generalization. Even if this problem is somehow overcome, there remains the issue that only so many predicates are attested in corpora such as SemCor, whose creation remains prohibitively expensive.

For these reasons, we introduce a new approach to adding relations to the KG. It is based on the idea that all predicates in the dictionary that can have arguments are potential nodes from which to extend new relations. Thus, one could try to combine all predicate nodes with all possible argument nodes in the dictionary, as long as there is available a reliable filter to separate semantically meaningful relations from nonsensical or trivial ones.

6.2.1 Learning Grammatical Role Embeddings from Parsed Corpora

General Procedure

Obtaining such a general filter that is able to rank all kinds of binary predicate-argument relations is apparently not feasible in terms of a rule-based approach. Instead, we have attempted to create a filter based on distributed representations of meaning, so that we can obtain, essentially, probabilistic interpretations over the

relevance of constructed relations. This filter must be able to give us information about what constitutes a prototypical argument of a specific type for a specific predicate, so that we can decide which arguments from the dictionary are good candidates. This also implies that we need comparable representations of argument prototypes and regular word senses, or synsets (the nodes in the KG). We have called the abstract representations of predicate arguments "grammatical role embeddings". In this work we have explored working only with grammatical roles of type *subj*, *dobj*, *iobj* (following the Universal Dependencies schema), but in principle such relations and roles can be defined and learned using a more comprehensive set of dependency relations or some other representation formalism (see Simov et al. (2018) for a discussion of Minimal Recursion Semantics as one such type of analysis that is compatible with the approach).

The following steps provide a description of the procedure for extending the KG:

1. Generate a large enough corpus that encodes grammatical roles together with representations of individual words (and potentially their sense IDs).
2. Train a VSM on the corpus, so that distributed representations are made available for both grammatical roles and regular lemmas and synsets. In the case when synset IDs are not directly available in the corpus, obtain the relevant embedding by averaging the vectors for the lemmas within the synset. Do the analogous thing for calculating grammatical roles for synsets – average the individual grammatical role vectors per all lemmas in the verb synset.
3. Use the dictionary for the KG to identify all predicates of interest which have associated arguments (grammatical role labels) in the generated corpus.
4. For each predicate and for each grammatical role type, put the predicate in a pair with all argument candidates from the dictionary. Then evaluate how close each argument is to the prototypical grammatical role for the predicate. Keep all pairs where the argument is within a predetermined distance from the protypical argument in the VSM space.

Here is a short example. Let us assume that for the first sense of "arrive", linked in WN with synset ID "02005948-v", there are attested examples in the parsed corpus of relations of type *subj* between this predicate and syntactically connected arguments. This prompts the algorithm to generate all possible combinations between the predicate and noun synsets found in the WN dictionary. Each such

generated combination is a relation between the predicate synset and a noun synset. The procedure then iterates over this list of relations. When it encounters a specific noun synset, for instance one of the senses of "group" (00031264-n), it takes the synset embedding learned from the corpus and compares it to the grammatical role embedding for that particular combination of predicate and role, in this case "SUBJ_arrive". If the cosine similarity between the two vectors is higher than a pre-specified threshold, the relation passes through the filter and is added to the list of new relations in the KG.

Training Data and Learning

The simultaneous learning of both grammatical role and lemma embeddings is made possible by the preprocessing of the training corpus. In the work cited, the WaCkypedia_EN corpus (Baroni et al., 2009) was used as a source of implicit syntactic knowledge, after it was parsed with the Stanford CoreNLP dependency parser. The parser was configured to use dependencies of type "collapsed-cc", so that dependencies between content words are obtained directly, collapsing any intermediary functional links. In addition to this, the option ensures that dependency relations are propagated to coordinated words across conjuncts. Therefore, the analysis of the sentence "The doctor operated on the patient and ate lunch" would result in the extraction of two *nsubj* relations: "doctor-operate" and "doctor-eat". All words are replaced with their lemmas and whenever a word is analyzed as an argument to a verbal predicate, it is replaced with a special token: ⟨argument_type⟩_⟨verb_lemma⟩. The example sentence would be processed into "The SUBJ_operate operate on the patient and eat lunch" and also "The SUBJ_eat operate on the patient and eat lunch" with regards to the subject argument; additional sentences would be generated per each of the grammatical roles found in the sentence, namely *IOBJ_operate* and *DOBJ_eat*. The preprocessing phase thus produces a larger number of sentences compared to the original ones.

In addition to the real text corpus (RTC), a pseudo corpus (PCWN) was generated via the method outlined in the previous section, so that the relations encoded into WN can be learned as well. All nodes referenced in the random walks produced by the generation procedure are converted to lemmas from the selected synsets. The RTC was used to learn embeddings on its own and also in conjunction with PCWN (through concatenation), the latter combination being denoted as RTCPCWN. PCWN cannot be used on its own, because it

supplies no information about syntactic dependencies and therefore the two types of embeddings (of lemmas and grammatical roles) cannot be situated in the same space. As in the previously reported experiments, the Word2Vec tool was used with the same settings (context window size set to 5 word, 7 iterations, 5 negative samples, threshold for the downsampling of frequent words fixed to 1e-7). The models that performed best on the similarity task were further selected for evaluation via KBWSD.

Initial experimentation with the outlined setup did not yield sizable improvements. A possible reason for this was identified in the fact that generating synset embeddings via the averaging of attendant lemma embeddings is very noisy, especially when the lemmas in question are ambiguous in terms of their POS category. In order to deal with this problem, two modifications to the training data were adopted. Firstly, the KBWSD approach described earlier was used to annotate the RTC corpus, so that the VSM could be trained directly on synset ID annotations; also, the nodes in the PCWN random walks were not converted to lemmas, but left as synset IDs. In this way, synset embeddings are obtained directly via training. However, this approach did not lead to improvements either, which is not surprising, if one bears in mind that even the best results with KBWSD on SemCor have accuracy of less than 70%; thus, one type of noise is replaced by another.

The second attempt to deal with the problem involves narrowing down lemma representation. By using the POS annotations in the RTC data, the lemmas for individual tokens were substituted with *lemma-POS* strings. In this way, the lemma embeddings are focused per POS type and therefore averaging over the lemma vectors in a synset should be somewhat more informative. The same thing was done for the PCWN data. This modification did result in an increase of WSD accuracy and therefore all results reported here are based on this type POS-sensitive representation.

Experiments and Results

Tables 6.2 and 6.3 present the central experiments of this line of research. The first table reports on KBWSD when using only the RTC for leaning a VSM; the second one is based on VSMs that utilize both the RTC and the PCWN data. WSD accuracy is measured against the test part of SemCor used in the previous chapter and a smaller data set named M13 SemeVal – originally used

for the Multilingual Word Sense Disambiguation task at SemEval 2013[6]. Each table reports a baseline accuracy for the KG that includes only the original WN relations (called here wn30) and accuracy scores for enriched KGs, where each one of those includes new relations obtained with different thresholds for the similarity filter. For instance, the graph wn30RTC40 is composed of the original WN relations and of the relations extended on the basis of the RTC corpus. Only relations with noun arguments related to the relevant grammatical roles via a cosine similarity measure of 0.4 or more are allowed in the extension.

Knowledge Graph	SemCor	M13 SemeVal
wn30	51.56	48.41
wn30RTC40	50.32	49.51
wn30RTC45	**52.60**	49.57
wn30RTC47	50.20	48.47
wn30RTC50	50.34	49.63
wn30RTC52	50.58	**51.88**
wn30RTC55	51.05	51.70
wn30RTC57	51.60	51.52

Table 6.2: Results on KBWSD with relations ranked by embeddings from a POS tagged real text corpus. The maximum improvement for SemCor is **1.04** and for M13 SemeVal is **3.47**.

Knowledge Graph	SemCor	M13 SemeVal
wn30	51.56	48.41
wn30RTCPCWN35	51.88	49.27
wn30RTCPCWN38	53.68	51.39
wn30RTCPCWN40	53.91	**51.45**
wn30RTCPCWN42	**54.33**	50.42
wn30RTCPCWN43	54.08	50.18
wn30RTCPCWN44	52.56	49.93

Table 6.3: Results on KBWSD with relations ranked by embeddings from a POS tagged real text corpus and pseudo corpus. The maximum improvement for SemCor is **2.77** and for M13 SemeVal is **3.04**.

The results show that accuracy varies depending on the corpus used for testing. SemCor, a more balanced resource, benefits from specific parametrizations of the threshold, while the smaller and less heterogeneous M13 SemeVal points to another optimal configuration. Similar differences can be observed with regards to the training corpus that is used. As is suggested by the tables, too permissive (i.e. low) a threshold results in noisy KGs that do not improve on the baseline,

[6] `https://www.cs.york.ac.uk/semeval-2013/task12/`

while setting it too narrowly could preclude enough meaningful information from being added to the KG.

Table 6.4 shows results for KGs constructed via reduced VSMs that reflect only more frequent syntagmatic relations. Verb arguments that occur fewer than ten times in the corpus are not considered when learning grammatical role embeddings and are therefore excluded from the procedure for relation generation. This modification leads to a significant increase in the results on the M13 SemeVal data set, while the results on SemCor decrease. Possibly, the smaller data set contains predominantly more popular word senses, which would explain the difference in accuracy with respect to the two corpora.

Knowledge Graph	SemCor	M13 SemeVal
wn30	51.56	48.41
wn30RTCPCWN10-34	**52.35**	51.39
wn30RTCPCWN10-35	50.64	**53.04**
wn30RTCPCWN10-36	50.25	50.72
wn30RTCPCWN10-40	50.49	49.45
wn30RTCPCWN10-45	51.15	49.27
wn30RTCPCWN10-50	51.45	48.29

Table 6.4: Results on KBWSD with relations extracted after less frequent grammatical role embeddings were removed from the VSM. The improvement for SemCor is **0.79** and for M13 SemeVal is **4.62**.

Curiously, varying the similarity threshold has somewhat different overall effects in the different cases. With some combinations of training and testing corpora the results follow a hump shape, i.e. there is a peak in accuracy somewhere in the middle ranges of threshold parametrization and results to the left and right decline. But in other cases (table 6.2 and the first column in table 6.4) the shape is more wave-like. This suggests that the new relations interact with the original semantic network (and among themselves) in complex ways. More work is necessary in order to establish an analytical approach of reasoning about these effects. However, the results presented here strongly suggest that such an approach to identifying meaningful new knowledge can be used productively to model the lexicon.

Chapter 7

Recurrent Neural Networks for Word Sense Disambiguation

This chapter presents supervised neural network architectures for WSD. This research builds on previous related work (introduced in detail in chapter 3), while offering several novel modifications. It also depends heavily on many of the ideas introduced in the previous chapters: the core of the NN architecture is adapted from the one described in the chapter on POS tagging with NNs; the additional input features that are used are obtained via the methods described in chapters 5 and 6; one of the NN architectures also relies crucially on the methods for embedding information from KGs. The chapter is based mostly on Popov (2017)[1]; however, the results presented here include reports of newer experiments as well. It is organized in three sections; the first one describes the two NN architectures used for WSD, the second one reports on the experimental evaluation and the last one provides a discussion.

7.1 Neural Network Architectures for WSD

Below I describe two architectures for doing supervised WSD. Both use recurrent neural networks with bi-directional LSTM layers. While the first one produces more traditional classifier-based models, the second, albeit mostly the same, takes a somewhat different approach, similar to context representation with RNNs. For

[1]Popov, A. (2017). Word Sense Disambiguation with Recurrent Neural Networks. In Proceedings of the Student Research Workshop Associated with RANLP 2017 (pp. 25-34).

easy referencing, I will call the first architecture – Architecture A, and the second one – Architecture B.

7.1.1 Direct Classification of Word Senses

Architecture A – Description

The first supervised NN architecture presented here is essentially the same as that outlined in chapter 4 and used for POS tagging. Figure 7.1 presents again in diagrammatic form the high-level description of the architecture. As with the POS tagging task, the input words are fed one at a time into the RNN. At that point they are already converted to integers so that embedding lookup can be performed within the network. The networks allows two embedding lookups to be performed per the same input; in such cases the obtained vectors are concatenated (following the observations of Goikoetxea et al. (2016) that this simple method gives surprisingly competitive results). The recurrent part of the network can have a varying amount of Bi-LSTM layers, which can be parametrized with regards to their size, initialization and amount of dropout regularization (Srivastava et al., 2014). For each input sequence, those states of the hidden layer are selected which correspond in the time series to the words that must be disambiguated (i.e. they have an entry in the lexicon). The forward and backward hidden states per word are concatenated and the output layer then performs linear transformations on the hidden representations. In this way the representations, which are of size *2 * hidden_layer_size*, are "stretched" to the size of the number of synsets for all lemmas attested in the training data (WordNet contains over 100,000 synsets but many of those are associated with lemmas that never appear in SemCor). Finally, a softmax layer calculates a probability distribution over the lexicon vector.

The final disambiguation decision is taken with respect to the probability mass concentrated in the positions of the vector dedicated to those synsets associated with the lemma under consideration. Lemmas with only one associated synset are directly disambiguated to that sense. The final decision-taking phase outside of the NN also has access to POS information about the input words, which is used to filter out some of the irrelevant synsets in advance. Whenever it encounters a lemma that is unfamiliar from the training data, the architecture falls back to the WN 1st sense heuristic.

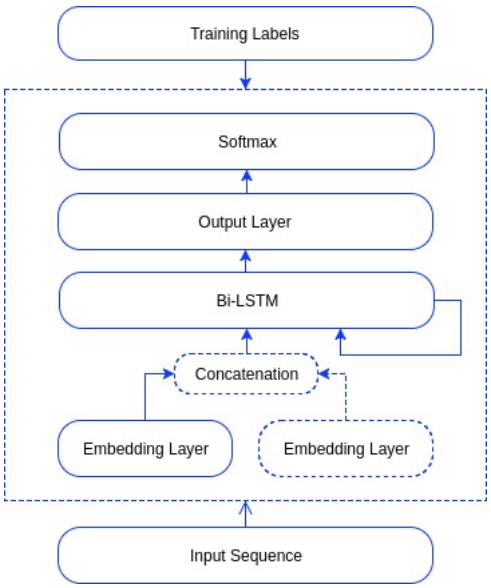

Figure 7.1: Recurrent neural network for word sense disambiguation: The dotted lines mean that a component or a connection is optional (in the case of concatenating embeddings from two different sources – e.g. word embeddings from natural text and lemma embeddings from a KG).

Input Features

The models trained with *Architecture A* do not use any hand-crafted features (though in principle nothing precludes that). Instead, all the information used to train on comes from the embeddings to which the input tokens are translated. The current work uses as principle features the word embeddings by Pennington et al. (2014) – GloVe. More specifically, it uses the set of embeddings trained on Wikipedia and the Gigaword corpus, or approximately 6 billion tokens of text, with a vocabulary of 400,000 uncased words and with dimensionality of 300. In the preliminary experimentation with popular freely-accessible embedding vectors this one gave the best performance and was therefore selected for further experiments. Exploring different VSMs and their parameters (e.g. what data they are trained on, the size of their vocabulary, method of training, dimensionality, etc.) remains an open task.

In addition to word embeddings, the network allows for concatenating a second embedding vector to the first one. This option was implemented so that the combination of word embeddings learned from natural text and lemma embeddings learned from KGs might be explored. In the section on results, some experiments are presented that test the effect of such combinations. The lemma embeddings are produced according to the procedures described in the previous

chapter. In particular, the VSM that, in combination with GloVe, consistently achieved the highest accuracy score on the development set is based on the already familiar KG combination **WN30WN30glConOne**. This model, when trained to predict contexts of 15 words, achieves some of the highest and best-balanced results on the two similarity and one relatedness datasets, which is why it was chosen for the experiments. It should be kept in mind that the lemma-centered VSMs supply meaningful information only with regards to content words and they are, naturally, unable to distinguish between word forms (this part of the architecture relies on input that has already been lemmatized).

7.1.2 Learning Lemma, Synset and Context Embeddings in a Shared Space

Architecture B – Description

The second proposed architecture is similar to the first one – except for the final stage where the representation of context is carried out and optimized and also with respect to how the input features are modeled with relation to the lexicon. Referring once again to figure 7.1, the formal differences between *Architectures A and B* are to be found *above the Bi-LSTM layer box*. They are two: 1) the output layer no longer maps the hidden contextualized representation of words to a lexicon-sized vector; here it maps the hidden layer output to a vector that is equal in size to the input embeddings of individual words, e.g. 300 dimensions; 2) the loss function that is being optimized is no longer *cross entropy*, but a *least squares* comparison between the context representation and a synset embedding for the gold label in the training data. For a visual representation, see 7.2

As outlined, the models trained by *Architecture B* are essentially doing the following:

1. A sequence of vectors, one per word/lemma in the input sequence, are fed into the hidden layer.
2. The Bi-LSTM layers produce a context representation per each word/lemma.
3. The context representations that correspond to open class words are selected.
4. Each context representation is resized to match the dimensionality of the input vectors.

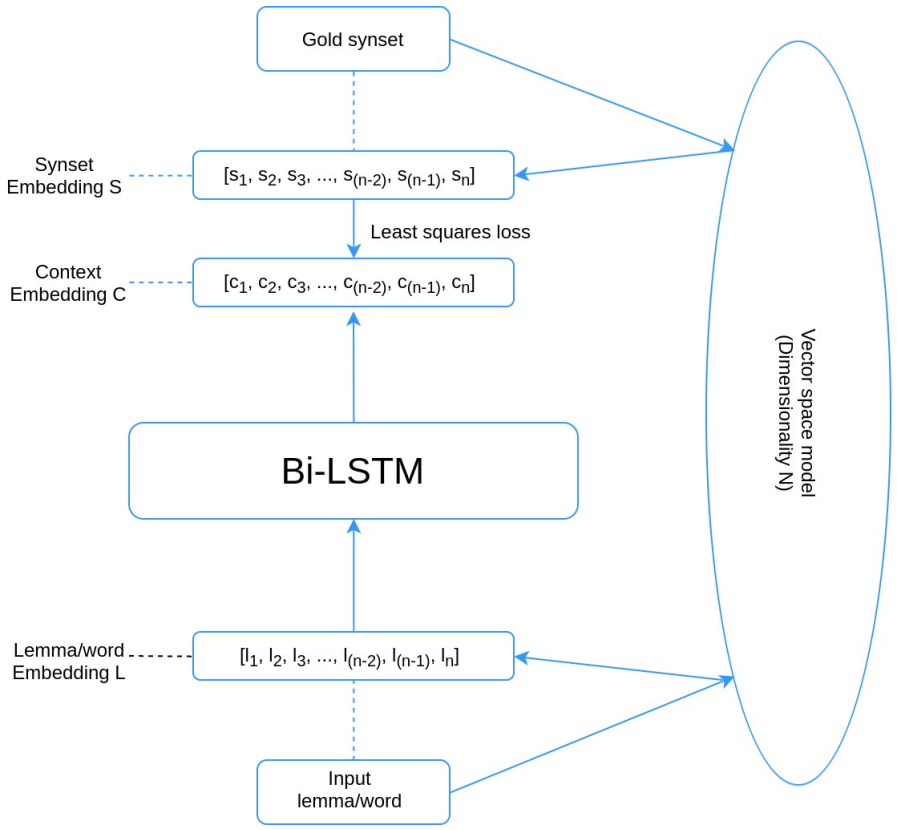

Figure 7.2: Diagrammatic representation of *Architecture B*. The same principles apply as with *Architecture A*, but the output layer produces a vector of the size of the VSM, which is then compared to the embedding vector for the gold synset; a mean of the least squares error is back-propagated as a learning signal. Crucial to the architecture is the availability of a VSM where both lemmas/words and synsets are represented as vectors of the same dimensionality.

5. The resized context representation is compared with a distributed representation of the corresponding gold label synset.
6. The network is optimized on the mismatch between the two vectors.

This approach relies on a VSM that can provide representations both for the input words/lemmas and for the gold label synsets, so that in effect the network learns to situate particular contexts in the embedding space. This permits the system to take the resultant context representation and to find the closest distributed representation of a synset linked to the word/lemma that is being considered for disambiguation.

Architecture A tries to pick just one position in the large lexicon vector at its end and to depress probability mass in all other dimensions. This means that it is exploring a single source of information from the gold data – "this is the correct

answer and everything else is wrong". *Architecture B* meanwhile aims at learning from a richer representation – by embedding the gold synset in a VSM, at least hypothetically it should have access to a range of semantic features that describe the correct answer, therefore it is a much more detailed model of the lexicon that provides an interpretation of the meaning of a word sense. Naturally, this method depends very much on the quality of the VSM used – in terms of the input representations (of words/lemmas), but also on that of the gold labels (synsets). As discussed in chapter 3, obtaining such mixed distributed representation models is an open task with no easy solution. One of the motivations for developing *Architecture B* is to demonstrate the potential utility of such resources. This approach has a distinct advantage, in that it can make decisions even for words it has never seen in the sense-labeled data – it only needs the relevant word/lemma and synset representations to be available in the VSM.

Input Features

Several "mixed" VSMs are used in the experiments with Architecture B. The methods outlined in chapter 6 are once again used in order to train distributed representation models that here combine information about lemmas and synsets in a shared embedding space. Producing such models involves a trivial change in the procedure for generating pseudo corpora – the algorithm is set to emit synset IDs in some cases (in this case, 50% of the time), and lemmas in all the rest. An additional method for combining lemmas and synsets is used as well – picking randomly a lemma from each referenced synset and inserting it next to the synset ID in the random walk (i.e. instead of substituting the synset).

More sophisticated methods of corpus construction are certainly possible – for instance, using the WN glosses in order to inject natural language patterns in between the nodes in the random walks – but this is left for future research. A naive strategy is attempted for enlarging the coverage of lemmas and thus including functional words, as well as to add more syntagmatic knowledge – a Wikipedia dump is lemmatized and concatenated to the pseudo corpus. In this way, some of the lemmas in WN occur both in the KG-generated random walks, together with synset IDs, and in natural language sentences, which allows the NN model to explore both types of knowledge, as well as to learn representations of words absent from WN (mostly function words) and to relate them to the synsets. The results demonstrate that even such a simple approach can lead to

big improvements, suggesting that models based on something like *Architecture B* could compete with more traditional classifier-like models.

7.2 Experiments and Results

7.2.1 Training and Evaluation Data

The training and evaluation of the proposed models was carried out within the Unified Evaluation Framework (UEF) by Raganato, Camacho-Collados, & Navigli (2017), so that the experimental results can be compared with those reported there[2]. All data in SemCor, as processed within the UEF[3], was used for training. The Senseval-2 data set was used for development and the rest of the evaluation data sets within the UEF – for the final evaluation. The differences between some of the models are evaluated solely on the development set at the intermediate stages, while the best models are evaluated on all data sets made available in the UEF.

7.2.2 Experimental Results

Table 7.1 presents the parameters for the model that achieved the highest accuracy on the Senseval-2 data set (called *Model A1*). The embedding size denotes the number of dimensions in a single VSM, so that the concatenation of word and lemma embeddings is double that size. The VSM from which the lemma embeddings are derived is based on the **WN30WN30glConOne** KG that has been described in the previous chapter. The training of the embeddings was done with the word2vec tool with the following parameters: number of training iterations = 7; number of negative samples = 5; window size = 15; threshold for the occurrence of words = 1e-7; algorithm = Skip-Gram. The training corpus is comprised of 500 million random walks on the KG, output by the UKB tool. The architecture trains for a fixed number of epochs, after which the best saved state on the development set is selected as final output of the training process.

[2]The data, results and descriptions of systems in the UEF are available at `http://lcl.uniroma1.it/wsdeval/`

[3]The data sets are POS-tagged and lemmatized with the Stanford CoreNLP tookit, available at `https://stanfordnlp.github.io/CoreNLP/`. Word senses are mapped to the WordNet 3.0 sense inventory.

Parameter	Value
Embedding size	300
Word embeddings	GloVe
Lemma embeddings	WN30WN30glConOne
Bi-LSTM hidden units	2 * 200
Bi-LSTM layers	1
Dropout	20%
Optimizer	SGD
Learning rate	0.2
Initialization of LSTMs	random uniform [-1;1]
Maximum training batch size	100
Training epochs	100000
Best result at training epoch №	58100

Table 7.1: Parameters for the *Architecture A* model with the highest accuracy on the development set.

Table 7.2 gives the parametrization of the most accurate model trained with *Architecture B* (*Model B1*). Unlike the model described in the previous table, this one does not use word embeddings as input, its hidden layers are two times larger, it has no dropout applied during training and its best result is produced much later (all models are stopped at 100,000 iterations).

Parameter	Value
Embedding size	300
Lemma embeddings	WN30WN30glConOne + WikiLemmatized
Bi-LSTM hidden units	2 * 400
Bi-LSTM layers	1
Dropout	0%
Optimizer	SGD
Learning rate	0.2
Initialization of LSTMs	random uniform [-1;1]
Training batch size	100
Maximum training epochs	100000
Best result at training epoch №	94200

Table 7.2: Parameters for the *Architecture B* model with the highest accuracy on the development set.

Table 7.3 presents a comparison of *Model A1* and *Model B1* with some of the systems evaluated in the UEF. It also gives the results achieved with two other models that are identical with *Model A1*, except for: *Model A2* uses only the GloVe word embeddings; *ModelA3* uses another set of lemma embeddings. The

VSM in *Model A3* is based on the WN30WN30glConOne KG as well, but also on the Wackypedia corpus discussed in the previous chapter with relation to the grammatical role embeddings. It is in practice the same corpus used to train the embeddings for grammatical roles and lemmas in the same space.

System	SNE-2	SNE-3	SME-07	SME-13	SME-15	ALL
IMS-s+emb	**72.2**	**70.4**	**62.6**	65.9	71.5	**69.6**
Context2Vec	71.8	69.1	61.3	65.6	**71.9**	69.0
Model A1	70.4	68.2	57.8	65.3	69.1	67.7
Model A2	69.6	69.4	59.3	65.0	69.4	67.8
Model A3	70.1	68.8	56.3	64.2	69.6	67.4
UKB-g*	68.8	66.1	53.0	**68.8**	70.3	67.3
IMS-2010	68.2	67.6	59.1	-	-	-
MFS	65.6	66.0	54.5	63.8	67.1	64.8
IMS-2016	63.4	68.2	57.8	-	-	-
Model B1	64.7	57.9	47.9	61.9	64.8	61.3
UKB-g	60.6	54.1	42.0	59.0	61.2	57.5

Table 7.3: Comparison of the models trained with *Architecture A & B* with other systems trained on SemCor and evaluated on several data sets ("SNE" stands for "Senseval", "SME" stands for "SemEval"). *IMS-s+emb*, *Context2Vec*, *UKB-g**. *UKB-g* and *MFS* are reported in Raganato, Camacho-Collados, & Navigli (2017); IMS-2010 is reported in Zhong & Ng (2010); IMS-2016 (this is the configuration IMS + Word2Vec (SemCor)) is reported in Iacobacci et al. (2016). The results from the UEF stand for the F-1 score, but since all systems there either use a back-off strategy or are knowledge-based, this is equivalent to accuracy, just as in the present work.

Models A1-3 are below the state of the art, but not by a large margin. They are comfortably ahead of the *Most frequent sense* (MFS) baseline, which is typically hard to beat; on some of the data sets the models perform better than the *UKB-g** configuration of the popular knowledge-based tool, beating it on the overall evaluation as well. *UKB-g** is itself much better than previous configurations of UKB, as shown in the table. Previous evaluations of the *It-makes-sense* system also perform below the three models.

Models A1 and A3 perform somewhat better than *Model A2* on some of the data sets, suggesting that the lemma embeddings do contribute relevant new knowledge, but they also fare worse on others, therefore the evidence is not unambiguous. A different model, identical to *Model A3* but trained at an earlier stage of experimentation, achieved 71.6% accuracy on the SNE-2 data set. Since the result could not be reliably replicated subsequently, it is not reported in the table. This does suggest, however, that with a better parametrization of the

learning algorithm, which is more stable with regards to learning, the combination of *Architecture A* and a WN-based VSM could achieve much higher accuracy that is almost equal to state-of-the-art results.

Finally, I present some results from experiments with different VSMs as sources of input features to *Architecture B*. The evaluation is done only with respect to the development set, Senseval-2, and the goal is to demonstrate how different approaches to distributed representation lead to different quality of the embedding. This is particularly important in the case of *Architecture B*, as the network effectively learns how to navigate within the same VSM – it takes as input word/lemma embeddings, calculates via recurrences the contexts for the input tokens and tries to match those to the corresponding synset representations *within* the same space. This means that the more meaningfully words/lemmas and synsets are related spatially (and therefore semantically), the easier it is for the network to establish how to map actual input to expected output.

Vector Space Model	Accuracy (SNE-2)
WN30WN30glConOne-C3 + WikiLemmatized	**64.7**
WN30WN30glConOne-C3	63.1
SW2V (600 hidden units)	62.1
WN30WN30glConOne-C1	61.7
SW2V (400 hidden units)	60.2
WN30WN30glConOne-C2	57.4
AutoExtend	53.2

Table 7.4: Comparison of the models trained with Architecture B on the Senseval-2 data. The parametrization of the models is the same (except for one of the SW2V models which has more hidden layer neurons). The SW2V embeddings are associated with mixed case strings of word forms as described in Mancini et al. (2016); the AutoExtend vectors are described in Rothe & Schütze (2015).

Table 7.4 shows that simply concatenating the lemmatized Wikipedia dump to the pseudo corpus leads to a big improvement – probably due both to the ability of the model to represent words missing in WN (mostly functional ones) and to having access to syntagmatic knowledge from natural language text. There is also difference in the performance of the models depending on how the pseudo corpus was constructed. Corpus1 (C1) was built by generating 100 million random walks from the graph and then adding next to each synset ID in the random walks a randomly chosen lemma from that synset; Corpus2 (C2) was built by generating 200 million random walks and directly substituting synset IDs with representative lemmas. Thus, C1 and C2 are roughly of the same size, but C1 is

much more effective in this evaluation. Corpus3 (C3) was built in the same way as C1, but the number of random walks in it is 200 million, i.e. twice bigger; it is the best-performing model based on pseudo sentences only. VSMs that also represent words/lemmas and synsets in a shared space, but are constructed differently (see chapter 3), like SW2V and AutoExtend, do not seem to be better than the simple approach proposed here. The SW2V embeddings, which are directly trained on natural language text (non-lemmatized at that), do perform a little better than some of the pseudo-copus-only-based embeddings (C1 and C2, bot not C3), if the hidden layer of the RNN is enlarged (which is not the case with the pseudo-text vectors); however, the combination of WN and Wikipedia beats all other VSMs, indicating that the KG is crucial for representing the relation between lemmas and synsets.

7.3 Discussion and Further Work

The results presented show that neural sequence architectures are a viable option for training supervised models for WSD. *Architecture A* – a recurrent neural network with a classifier on top – is able to achieve results which are relatively close to the state of the art compared to other approaches, without the need for heavy feature engineering (*IMS*) or complex pre-training procedures (*Context2Vec*). The results are not fully comparable with those in the UEF, as this work does not use the same VSMs (e.g. the size of the vectors used there is 400, compared to the 300-position vectors I use in this thesis), but the rough comparison shows that RNNs can compete with the best systems, especially if more effort is put into optimizing the parameters through heavy experimentation. Some initial results indicate that even with the currently identified optimal parameters, much higher results are possible, even though difficult to replicate. Therefore, further work is necessary to find more stable learning configurations. Additional improvements, like for instance adding a CRF layer on top, so that decisions are taken with respect to the global well-formedness of the semantic analysis, could potentially make the architectures even more competitive.

Architecture B, while not as accurate in the experimental evaluation, shows potential for an alternative approach to WSD with RNNs. Under this approach, the system attempts to learn how to navigate the dynamics of meaning – how to start from representations of individual words/lemmas and to combine them

in contextual representations that are close to pre-learned representations of concepts/synsets. Such a resource, i.e. a mixed VSM of lemmas, words and synsets, could be useful in other applications, not merely in WSD and sequence-to-sequence tagging tasks, e.g. as a means to identify semantic axes of variation between senses, between senses and words, etc. The experimental results indicate strongly that there is a lot of space for improving such VSMs, since even straightforward modifications produce big gains in accuracy.

In the next chapter I show how the combination of the two approaches (*Architecture A & B*) can also have useful effects, therefore suggesting that the two methods learn slightly different kinds of linguistic meaning and can complement each other.

Chapter 8

Multi-task Learning with Recurrent Neural Networks

The final chapter in the thesis presenting original research seeks to explore the possibilities of doing WSD in parallel with other NLP tasks. This line of work is important in order to motivate further research in WSD, which has not been definitively shown to benefit downstream applications. This big detriment in the field of WSD is due partly to the fact that proper sense-annotated data is very expensive to produce and consequently no large-scale projects have attempted to exhaustively explore the value WSD can bring into general NLP. From a theoretical point of view, however, lexical semantics is of tremendous interest, as, according to many theories and formalisms, it is in the lexicon where much of the structural patterning in language is encoded: from syntactic valency frames and semantic frames of various kinds to morphological characteristics, selectional restrictions and discourse information.

RNNs offer a good opportunity to explore this hypothesis (that the lexicon is an interface between different types of linguistic knowledge) – especially via *multi-task learning*. This term denotes the process of training a system to solve two or more tasks in parallel, so that the separate classifiers (or other kinds of decision modules) make independent decisions but also share hidden parameters and all contribute collectively to their optimization in training. This approach is very different from the standard pipeline solutions, where each processing module works on its own and then feeds its annotations to the following one in the chain. Errors propagate rapidly along complex pipeline architectures and consequently modern research is moving more and more towards a multi-task learning orientation

where the separate tasks are solved in parallel and the relevant modules share the same constraints. This sharing of parameters has the potential to harmonize the different kinds of analyses and to exploit information from various levels of linguistic structure, in a more holistic manner.

The chapter outlines two examples of multi-task learning. The first one combines *Architectures A and B* from chapter 7 and the second one seeks to solve in parallel WSD and POS tagging. Positive results in these preliminary studies should provide incentive to pursue further the serious development of WSD models and their integration in larger systems that seek to model language more extensively. In such an endeavor, modeling the lexicon should be a central task that provides firm foundation to almost everything else.

8.1 Combining a WSD Classier and a Learner of Context Embeddings

The first idea for multi-task learning that is explored here combines *Architectures A and B* from the previous chapter. The two methods share the same hidden layer(s), but have their own, parallel output layers where the training signal is obtained. The motivation to do this kind of parameter sharing is that even though both *Architectures A and B* are used to perform WSD, they actually solve tasks which differ significantly. *Architecture A* aims to directly pick the most relevant word sense out of the full vocabulary of available senses and therefore uses a large lexicon vector to represent its beliefs about the most probable choices. Rather than doing direct classification, *Architecture B* attempts to represent the sentential context with regards to the word to be disambiguated – as a vector within the original embedding space that provides the input features. The working hypothesis is that even though the two methods use a lot of common information, their different objectives force them into learning different kinds of lexical knowledge as well – perhaps more targeted with respect to *Architecture A*, and a more detailed but also multi-purpose representation in the case of *Architecture B*.

Therefore the changes to the implementation are minimal. In fact, there are none except that both types of hidden-to-output layer connections are present (one giving a vocabulary-sized vector, the other an embedding-sized one) and that the results of the two cost functions (cross entropy and least squares) are

summed to give a final number to the optimizer. The same parameters for the RNN are used as those in table 7.2 from the previous chapter, as well as the same embeddings (**WN30WN30glConOne-C3 + WikiLemmatized**). Table 8.1 provides results on all evaluation data sets, compared with the results for the single-task architectures trained with the analogous parameters.

	System	SNE-2	SNE-3	SME-07	SME-13	SME-15	ALL
similarity	Model B1 (single)	64.7	57.9	47.9	61.9	64.8	61.3
	Model B1 (multi)	66.8	60.1	49.2	63.4	67.7	63.3
classification	Model A1 (single)	70.4	68.2	57.8	65.3	69.1	67.7
	Model A2 (single)	69.6	69.4	59.3	65.0	69.4	67.8
	Model A3 (single)	70.1	68.8	56.3	64.2	69.6	67.4
	Model A4 (single-200)	67.7	66.9	55.8	63.6	68.3	65.9
	Model A4 (single-400)	68.5	67.1	58.2	63.6	67.0	66.2
	Model A4 (multi)	68.9	67.8	58.0	63.7	68.4	66.7
	Model A4 (multi+dropout)	69.6	68.0	59.1	64.5	70.2	67.5
	MFS	65.6	66.0	54.5	63.8	67.1	64.8

Table 8.1: Comparison of single-task and multi-task models. The first section of the table presents accuracy results on all evaluation data sets for two similarity (*Architecture B*) models that are initialized with the same parameters and embedding vectors (see chapter 7 for details); the only difference is that one has been trained only on the similarity task and the other one has been trained together with an *Architecture A* type classification model.
The second section presents accuracy results for classification models. The results for *Models A1-A3* are repeated from chapter 7. The *Model A4* variations have the same parametrizations and inputs as *Model B1 (multi)* with regards to their shared components; the only exception is Model A4 (single-200), which has half the number of hidden units. This is motivated by the better performance of *Architecture A*-type models with smaller hidden layers – when the GloVe vectors are used. *Model A4 (multi)* shares the same principles and parameters with *B1 (multi)* and is in fact trained together with a B-type model, i.e. *A4 (multi)* and *B1 (multi)* are just the two separate pathways of one and the same model.

The results demonstrate that multi-task learning does help in this case. The multi-task models (*A & B*) are both more accurate than their single-task counterparts. In the similarity-style WSD (*Architecture B* type models) the difference is greater: it ranges between 1.3% and 2.9% accuracy on the different data sets, with a 2% overall difference. *Model B1 (multi)* is able to overcome the MFS baseline on two data sets (SNE-2 and SME-15) and is only 1.5% below it on the overall evaluation.

The classification model (*Architecture A* type) that shares parameters with a similarity-based module also performs better than its purely classifier-based analogous versions (*A4 (multi)* as opposed to *A4 (single)*), especially when dropout on the hidden layer is added (the amount of regularization used in the model referenced in the table is *0.5*). Note that in addition to a 400-hidden-units single-task model, one with 200 units was trained as well, since in the case when using the GloVe embeddings this parameter setting gives higher accuracy for this type of architecture. Moreover, *Model A4 (multi*) is able to score relatively closely (the dropout model actually does better on some data sets) to Models A1-3, which all use the GloVe vectors, i.e. they have access to a VSM with much more accurate representations of the input words. It is noteworthy that the GloVe vectors represent word forms, whereas the VSM used by *Models A4* encodes representations of lemmas, i.e. it doesn't make any use of morphological information.

These results are encouraging because they suggest that: 1) there is a significant amount of mutual support between the two tasks; 2) the poverty of the graph-induced vectors (compared to the GloVe vectors) can be somewhat mitigated in such multi-task learning settings.

8.1.1 Analysis of the Results

Here I offer an analysis of the behavior of the two subsystems in the multi-task learning model, in order to demonstrate that the A and B branches learn different types of information and it is not the case that they give the same answers, with one of them being just a little more accurate. To this purpose, three subsets of the gold annotations were excerpted from the *ALL* evaluation data set, together with the corresponding answers given by: the classification module (here called *A*), the similarity module (here called *B*) and the WordNet 1st sense heuristic (here called C). The excerpted annotations all correspond to three types of situations. For obvious reasons, we are not interested in the cases where A and B provide the same answer, so this leaves the following: 1) A, B and C all give different answers; 2) B and C give the same answer, which is different from that provided by A; 3) A and C give the same answer, which is different from that provided by B. Table 8.2 provides an overview of how often one or the other model is correct.

If it were the case that the type B modules (similarity) are merely learning the same information as type A modules (classification), one would expect to find

Combination	A!=C!=B	B=C!=A	A=C!=B	Total
A correct	46	256	452	754
B correct	79	598	257	934
C correct	78	598	452	1128
Neither correct	82	229	241	552
Both (A&B) correct	3	12	15	30

Table 8.2: Comparison of different models. The first column gives information about cases where neither of the three models agrees with any of the rest; in the second column the similarity module picks the same answer as the WN 1st sense heuristic; and in the third one the classification module conforms to the WNFS heuristic. "A" stands for "classification module"; "B" – for "similarity module"; "C" – for "WN 1st sense". The "Both correct" line means that the two modules (A and B) chose different synsets which are both listed in the gold annotation.

almost no examples where the similarity module, and not the classification one, provides a correct answer, especially when its answers deviate from the WN 1st sense heuristic, which in a way corresponds to the MFS heuristic and is something that can be learned from the training data fairly well. On the contrary, the similarity architecture knows better than the classifier in many cases. In fact, it is more often correct in its predictions, by a wide margin. This does not belie the higher accuracy score of the classifier approach in general, as the latter uses a backoff heuristic to the WN 1st sense whenever it encounters a word it has not trained on. But if such cases are counted as errors on the part of the A models, then the similarity module is clearly more powerful than the softmax-based part of the architecture. Model B is also leading the board in the case where all three models give different answers (albeit it is in practice tied with model C). And when the classifier and the 1st sense heuristic are in agreement, the similarity module is correct in about a quarter of all cases. This short analysis is of course far from sufficient for any final conclusions, but it nevertheless strongly suggests that the two pathways in the multi-task learning architecture indeed pick on different kinds of data. Therefore figuring out how to integrate them even better and how to build ensemble models for combining their answers might lead to further improvements with respect to WSD.

8.2 Combining POS Tagging and WSD

Secondly, another multi-task learning setup is explored briefly. This scenario brings together two tasks I have discussed in different chapters of the thesis: WSD disambiguation and POS tagging. Whereas the previous example of multi-task learning combines modeling objectives that are, at least conceptually, relatively close to one another, here two very different (albeit related) aspects of the lexicon are modeled in parallel: morphological and lexico-semantic knowledge. Again, the same basic architecture is used; this time the two diverging hidden-to-output paths are analogous to one another – they map the context representation to the size of the tag set and calculate a probability distribution using a softmax function. Thus, there are two such parallel pathways: one for WSD and one for POS tagging.

The training data for the models is again SemCor, but this time the original POS annotations are used[1]. The POS tag set is converted to the same coarse-grained labels (Petrov et al., 2011)[2] used in the Universal Evaluation Framework: it comprises of just 12 broad categories (e.g. NOUN, VERB, ADVERB, etc.), which is useful for this experiment, since its purpose is not to achieve state-of-the-art results, but to investigate the potential interaction between the two types of tasks. The development corpus, Senseval-2, is used for evaluation purposes. The original Senseval-2 corpus does not include POS annotations, but the UEF version is manually corrected with respect to POS tags for all content words; this, in combination with the coarse-grained nature of the tag set, should result in a relatively low degree of errors, so that the corpus can be thought of at least as a kind of "silver" resource. Note that the results for WSD reported below are somewhat higher that those in the previous chapter. This probably has to do with differences in the training data, but since I am using this different version only for the sake of the gold POS annotations, I will not be analyzing possible divergences between the files (the main motivation for using the UEF corpora in the first place is so that a comparison with other WSD systems can be easily carried out).

Table 8.3 presents the exploratory results of this first foray into combining WSD and POS tagging in an RNN setup. The parametrization used is in practice

[1]Downloaded from `https://github.com/rubenIzquierdo/wsd_corpora/tree/master/semcor3.0`

[2]`https://github.com/slavpetrov/universal-pos-tags`

System	WSD (SNE-2)	POS (SNE-2)
Model A2 (single-WSD)	70.6	-
Model A2 (single-POS)	-	90.9
Model A2 (multi)	**71.1**	**92.1**

Table 8.3: Comparison of single-task models that learn to solve only either WSD or POS tagging, and a multi-task model that learns to solve both in parallel. "SNE-2" stands for "Senseval-2".

the same as that for *Model A2* from the previous chapter, including the embedding layer for which the GloVe vectors are used. Therefore *Model A2 (single-WSD)* is in practice the same model, while the other two differ from it with respect to their changed objective functions. For the sake of simplicity, I designate all three as some version of *Model A2*. The accuracies in the table show that the multi-task model does indeed fare better both with respect to WSD and POS tagging, more significantly in the latter task.

8.3 Discussion and Further Explorations

The two experiments discussed in the chapter suggest that multi-task learning can indeed be used to push current results forward. In the first example, the capability to select one single correct word sense is shown to work in complementary ways with the capability to translate the context of a word into a meaning representation within the same VSM used for the representation of the inputs and the gold labels. The second experiment shows that there is some interaction between morphological and lexico-semantic patterns.

Further explorations of multi-task learning setups are certainly necessary in order to determine which tasks benefit most from co-training with WSD models, and which tasks help WSD in turn. POS tagging, for instance, does not seem like an ideal candidate from this point of view, as morphological patterning seems to be much more co-dependent with syntactic structure. Syntactic and semantic valency analysis should, however, be very good sources of complementary data that is nevertheless crucially dependent on knowledge of the lexicon. The only reason POS tagging was selected for this demonstration is that the implementation of the system is much easier. Other, auxiliary tasks could also help drive accuracy upward. Several such tasks were tried out before writing the thesis, such as

attempting to guess the hypernyms of words and learning the distinction between frequent and rare words, but none of them produced interesting results. However, more experimental work is necessary in order to determine which auxiliary tasks do and do not help with WSD (or other problems). A unified solution that is able to model language in many different ways, while sharing most of its parameters amongst the kinds of analyses it produces, would be a serious step towards building multi-purpose and complexly structured linguistic and conceptual representations that resemble human thought, rather than task-specific machinery.

Chapter 9

Summary and Outlook

This chapter serves as a summary to the thesis and provides a conclusion to it as well. It also discusses the outlook for further work related to the lexical modeling approaches presented here.

9.1 Summary

Representing lexical information has been an important task in the fields of NLP and computational linguistics from their inception. Early work in machine translation was especially focused on identifying the correct lexical senses of words in order to provide precise lexical mappings between languages. The lexicon was given a central role in formalisms like *Head-driven phrase structure grammar* and *Lexical functional grammar*, which have been used in computational modeling of language – in that it controls the syntactic and semantic patters lexical items can combine into and command. The lexicon can also include information about the discourse usages of its member items, stylistic and socio-linguistic information, etc.

However, a number of obstacles have impeded substantial progress in the area of computational lexical modeling. First of all, the sheer difficulty involved in doing work at a number of levels related to lexical modeling turned out to be a significant barrier. These difficulties include: organizing the lexicon in a way that provides sufficient information but does not partition linguistic concepts far too granularly; representing meaning in terms either of features or of relations between the lexical items (i.e. constructing a semantic network); performing

word sense disambiguation in order to identify specific lexical items in text. The enumerated quandaries are not separate from each other, they are all significantly interconnected, so that, for instance, a particular level of granularity of the lexicon implies a different way of representing meaning and will make WSD more or less difficult. Further difficulties are connected with the more directly practical side of things: constructing lexicons is very slow and expensive and specific to different languages; training accurate WSD models requires a lot of data, since each lexical item in practice presents a separate classification task, and it is difficult to provide corpora with exhaustive coverage of the lexicon; the annotation process itself is very slow and expensive, much more so than doing POS tagging, for instance. Last but not least, the benefits of lexical modeling in relation to other NLP tasks remain uncertain. There has been little confirmation of the positive effects that WSD can bring to downstream tasks – be they in terms of additional features for supervised models, as a source for rule-based decisions, etc.

All reasons listed above contribute to some extent to this lack of experimental confirmation of the value of performing WSD and related lexical analyses, with regards to computational tasks. The poverty of training data (where WSD annotations are combined with data that can be used for other NLP tasks) is central among these, since it makes testing the hypothesis ("WSD is important for downstream NLP tasks") almost impossible. This precludes constructing easily interpretable scenarios with gold data that can directly show whether WSD can help other tasks or not. This will probably remain true at least until high enough accuracy scores are achieved with respect to WSD, so that such modules can provide reliable information to related language processors. But even more fundamental questions remain open-ended with respect to lexical modeling. For instance, if the lexicon potentially stores information that connects together various levels of linguistic analysis, how should that wealth of meaning be organized so that it can be used to structure analyses that go beyond merely identifying a relevant word sense? Even if accurate enough WSD is achieved, how should other NLP modules interpret this kind of information, what kinds of interfaces should be used between the modules? In general, what should be meant by lexical modeling in the context of NLP – does it include only the lexical-semantic aspect of meaning, or does it also include more abstract things, such as morphological and syntactic information, sentential semantics, etc? And last but not least, should that information be encoded in the form of symbolic structures or as probabilistic preferences, or should there be a hybridized approach that combines the two kinds

of encoding? What kinds of computational models are then most suitable to doing lexical analysis and can the strengths of different computational approaches be combined?

In this thesis I have attempted to address several of these questions and their interactions. It has taken up the most popular natural language lexicon in use in NLP (WordNet) and explored it from several points of view. In addition to using WN as an enumerative resource for word senses in the context of WSD, it has attempted to enrich it with additional information about lexical items. The kinds of enrichment proposed here are derived from the structure of WN itself, but also from related resources like annotated corpora (eXtended WordNet, SemCor, BulTreeBank) and the logic forms of WN glosses. The thesis has demonstrated that making the semantic network denser in terms of information-rich relations helps improve knowledge-based methods for WSD (on Bulgarian and English data). Continuing this line of thinking, it has explored a method for generating artificial corpora out of the enriched knowledge graph and consequently encoding this information in distributed representations of words, lemmas and synsets. This in some sense is conceptualized as a step toward encoding the symbolic information in WordNet (understood as the grouping together of senses, the symbolic nature of the relations between synsets and senses, the gloss definition in the synsets) into numerical data upon which computational operations can be performed. Such representations are shown to be highly-competitive on tasks that tap into lexical-semantic knowledge (word similarity and relatedness). Embedding methods are also used to generate distributed representations of a somewhat different kinds – of prototypical grammatical arguments of predicates. The latter are used as filters for the well-formedness (or plausibility) of relations in an experimental setup that attempts to enrich the WN network by exhaustive combination of noun and verb senses that have not been connected via syntagmatic links. This method is shown to yield improvements on the WSD task, which suggests that it could be worth the effort to explore it more comprehensively and apply it to uncover all kinds of possible new relations in the network. Together these approaches provide one way to combat the lack of enough training data, since they offer a way to infer lexical information that has not been attested in annotated corpora, and they also allow for enriching the lexical representation itself.

The thesis has also explored supervised methods for lexical analysis, more specifically such based on recurrent neural networks. The basis for the RNN architecture used throughout is first introduced in the context of POS tagging –

a task that is traditionally not thought of as a kind of lexical analysis, but that nevertheless relies on information that could be part of a lexicon representation (namely morphological information and knowledge about co-occurrence patterns and syntactic dependencies). That work has also been used to train distributed representations of words and morphological suffixes for the Bulgarian language, which were used as input features to the recurrent network. The architecture has been validated through relatively good evaluation results on the BulTree-Bank corpus, and the suffix embeddings have also been shown to contribute new information to the tagging process.

In the final part the thesis I have combined the separate strands of work in the preceding chapters in order to offer an increasingly integrated view on lexical modeling. First, the RNN architecture has been adapted to the task of WSD. A more traditional interpretation of the task (called *Architecture A*) has been implemented that directly classifies words into senses, based on a probability distribution on the output of the network and on a filtering outside the network that relies on the WordNet dictionary. Several models trained with *Architecture A* have been evaluated on a number of popular data sets for the English language; the models achieve results that are close to, even though not yet on par, with state-of-the-art results. Some of the models combine popular word embeddings (GloVe) with lemma embeddings generated via the previously introduced WN-based methods and in some cases these additional features do contribute to more accurate WSD. Additionally, a second architecture is implemented that, instead of directly attempting to choose the correct word sense, learns to represent the context of a word usage in a distributed manner. *Architecture B* thus is able to situate contexts in the same vector space model which also contains representations of lemmas and synsets (the VSM is again obtained via the method outlined earlier in the thesis). The best such model achieves lower results that the more traditional classifier-based method, and is not able to beat the most frequent sense baseline either, but it nevertheless comes close to that challenging threshold. It is important to note that models produced by *Architecture B* have the advantage of practically always being able to make a guess regarding the disambiguation of a word, while *Architecture A* models and other state-of-the-art systems necessarily rely on a powerful WN-based heuristic (choosing the first sense for a lemma, since it is carefully chosen by professional lexicographers as the most popular one).

Finally, the thesis attempts to combine the different kinds of analyses and representations that have been explored toward more unified models of lexical

representation. This is done within the paradigm of *multi-task* learning, i.e. by combining several different tasks that share common parameters but produce different training signals, which in the case of neural networks are backpropagated to the shared hidden layer. Such a shared representation should be learning to model linguistic input by paying attention to different aspects of lexical information, which may hold different degrees of relevance for separate tasks, but are nevertheless connected in ways that are perhaps not directly observable by single-task methods. Two cases are presented. The first one combines *Architecture A* and *Architecture B* in a single system that can perform both procedures in parallel. The evaluation results point to very significant increases of accuracy in both types of modeling – making *Architecture-A*-style modeling based purely on lemma embeddings almost on par with the model using the powerful GloVe embeddings, and boosting the *Architecture-B*-style module beyond the MFS baseline on some data sets. The analysis of the choices taken by the two computational pathways suggests that the they do in fact learn different disambiguation and representation strategies and as such are a good fit for a multi-task scenario. The second cases combines an *Architecture-A*-style module with a POS tagger like the one developed earlier in the thesis. The evaluation results strongly suggest that multi-task learning benefits both WSD and POS tagging, which would mean that these two aspects of modeling words do indeed interact and it makes sense to interface such representations in a computational lexicon.

9.1.1 List of Publications Related to the Thesis

This section provides a summary of the publications the author has worked on and of their relation to the parts of the thesis. With the exception of the final chapter on multi-task learning, all original work described in the thesis has been presented at international conferences and workshops and subsequently published in conference proceedings or journals. Combining these strands of work here is meant to provide a more global perspective to the connections that can be found in the separate approaches to modeling lexical knowledge.

No.	Publication	Summary	Ref. to Thesis Chapter
1	Popov, A. "Neural Network Models for Word Sense Disambiguation: An Overview." Cybernetics and Information Technologies 18.1 (2018): 139-151.	This journal paper provides an overview of different neural network models used to perform word sense disambiguation. It summarizes various neural architectures used for training language models, for representing linguistics units and contexts in a distributed manner, and for direct word sense classification.	3
2	Simov, K., Popov, A., Simova, I., & Osenova, P. (2018). Grammatical Role Embeddings for Enhancements of Relation Density in the Princeton Wordnet. In Proceedings of the 9th Global Wordnet Conference.	This paper, accepted to the global conference on WordNet and its versions and applications, presents a new approach to enriching the semantic network of the lexical resource by iteratively trying out potential combinations of noun-verb relations and selecting those that are automatically evaluated as plausible from a syntactic point of view. The method for filtering out implausible relations is based on a distributed representation model that can provide vector interpretations of lemmas, synsets and, what is new here, *grammatical role embeddings*, i.e. this work is able to provide a prediction of what good candidates for predicate arguments are like. In this way a semantic network like WordNet can be made much denser and indicative of world knowledge encoded in real text.	6

3	Popov, A. (2017). Word Sense Disambiguation with Recurrent Neural Networks. In Proceedings of the Student Research Workshop Associated with RANLP 2017 (pp. 25-34).	This paper presents two architectures for word sense disambiguation. Both architectures are recurrent neural networks (more specifically, LSTM cells are used for the recurrence). One architecture does more traditional, direct classification of the word senses and is able to achieve results approaching the state of the art; it also utilizes additional distributed representations based on the structure of the WordNet semantic network, which are shown to increase accuracy on the particular evaluation data set that is used. The second type of architecture attempts to represent word usage contexts in terms of a vector space model that is being used also for representing input lemmas and gold label synsets as numerical features; this system achieves lower scores, but the initial results are deemed enough to encourage further research.	7
4	Simov, K., Osenova, P., & Popov, A. (2017). Comparison of Word Embeddings from Different Knowledge Graphs. In International Conference on Language, Data and Knowledge (pp. 213-221).	This work compares the performance of different vector space models, when evaluated on the word similarity and relatedness tasks. Its main contribution lies in demonstrating that models produced on the basis of WordNet, and especially on enriched versions of WordNet, can be very competitive and actually better than models trained on data that does not explicitly encode lexical-semantic information. Popular word embedding models trained on large amounts of regular natural language text or on sequences derived from syntactic dependency trees are used as a baseline, as well as another model generated on the basis of the original WordNet network. The models presented in the paper are able to outperform the baselines, in some cases by a wide margin.	6
5	Popov, A. (2016). Neural Network Language Models–an Overview. In The Workshop on Deep Language Processing for Quality Machine Translation (DeepLP4QMT) (p. 20-26).	This workshop paper is an overview of various language models learned with neural networks: ranging from complex recurrent ones, through feedforward networks, to simple shallow models which allow for training on large amounts of natural language text and consequently for extracting high-quality word embedding models from the networks.	3

6	Simov, K., Popov, A., Zlatkov, L., & Kotuzov, N. (2016). Transfer of Deep Linguistic Knowledge in a Hybrid Machine Translation System. In The Workshop on Deep Language Processing for Quality Machine Translation (DeepLP4QMT) (p. 27-33).	This work explores a hybrid architecture for machine translation that combines a standard probabilistic model with post-processing rules for the transfer of information from source to target language. The transfer is carried via the Robust Minimal Recursion Semantics formalism, but the general approach presents general support for using lexical-semantic representations in machine translation, as the latter is shown to improve the score of the probabilistic translator.	3
7	Simov, K., Osenova, P., & Popov, A. (2016b). Using Context Information for Knowledge-based Word Sense Disambiguation. In International Conference on Articial Intelligence: Methodology, Systems, and Applications (pp. 130-139).	This paper outlines different strategies for enriching the WordNet semantic network, using other linguistic resources. The various combinations of new relation sets are evaluated via a knowledge-based word sense disambiguation method and against a baseline graph – on the SemCor data for English that uses the original WordNet relations and those derived from the WordNet annotated gloss corpus. The different strategies include extracting relations from the logical form of the WordNet glosses (included in the eXtended WordNet corpus) and from semantically annotated corpora (SemCor), following two different approaches to representing sentence structure. The best performing new graphs beat the baseline with more than 6% in terms of accuracy.	5
8	Popov, A. (2016). Deep Learning Architecture for Part-of-speech Tagging with Word and Suffix Embeddings. In International Conference on Artificial Intelligence: Methodology, Systems, and Applications (pp. 68-77).	The paper describes the implementation of a recurrent neural network architecture for part-of-speech tagging, trained and evaluated on data for the Bulgarian language (using a medium-grained tag set of 153 labels). It also describes the process of training vector space models for Bulgarian word forms and for quasi-morphological suffixes. The trained models all achieve accuracy above 90%, while the addition of the suffix embeddings as features leads to an increase of about 3%.	4

9	Simov, K., Osenova, P., & Popov, A. (2016a). Towards Semantic-based Hybrid Machine Translation Between Bulgarian and English. In Proceedings of the 2nd Workshop on Semantics-Driven Machine Translation (SedMT 2016).	This work describes a hybrid approach to machine translation, wherein a statistical machine translation system (Moses) is provided with lemma suggestions for word-level translation on the side of the target language and in the form of factors (features). The factors are generated via mapping the disambiguated senses of the source language words to their target language correspondences in a parallel WordNet. Some rules are used in the generation as well. The empirical results suggest that such a usage of word sense disambiguation might be able to boost statistical machine translation results.	3
10	Simov, K., Popov, A., & Osenova, P. (2016a). Knowledge Graph Extension for Word Sense Annotation. In Innovative Approaches and Solutions in Advanced Intelligent Systems (pp. 151-166). Springer.	In this paper a detailed analysis is carried out with regards the subsets of relations in the WordNet semantic network (including the gloss-derived relations from eXtended WordNet) and their influence over a popular graph algorithm for knowledge-based word sense disambiguation. It is argued that via such an analysis some of the relation sets that do not contribute meaningfully to the accuracy of the algorithm can be excluded from the graph. Such an analysis is also carried out with regards to new relation sets that are generated either via inference of the already present relations, or from syntactic parses in sense-annotated corpora for Bulgarian and English. The best graphs based on combinations of old and new relations significantly outperform the baselines which use the original WordNet network and gloss-derived network.	5
11	Simov, K., Popov, A., & Osenova, P. (2016b). The Role of the WordNet Relations in the Knowledge-based Word Sense Disambiguation Task. In Proceedings of Eighth Global WordNet Conference (pp. 391-398).	Another study that presents an in-depth analysis of the impact of the various subsets of relations within WordNet and eXtended WordNet, as well as of subsets of inferred relation sets, also analyzed by morpho-syntactic types.	5

12	Simov, K., Popov, A., & Osenova, P. (2015). Improving Word Sense Disambiguation with Linguistic Knowledge from a Sense Annotated Treebank. In Proceedings of the International Conference Recent Advances in Natural Language Processing (pp. 596-603).	This is the first paper in a sequence of studies that explore how a semantic network like WordNet can be expanded for the sake of improving knowledge-based word sense disambiguation. It deals exclusively with data for the Bulgarian language (using the BulTreeBank corpus for relation extraction and evaluation of the disambiguation algorithm) and relies on a partial mapping of the Princeton WordNet for English to Bulgarian word senses, preserving the English hierarchy and semantic and lexical relations. Different strategies for relation enrichment are explored, including various kinds of hypernymy inference, domain information inference and syntactic relation inference via an annotated corpus. The study reports a big improvement of knowledge-based word sense disambiguation accuracy on the available Bulgarian data.	5
13	Popov, A., Kancheva, S., Manova, S., Radev, I., Simov, K., & Osenova, P. (2014). The Sense Annotation of Bultreebank. Proceedings of TLT13, 127-136.	This paper describes the process of annotating the BulTreeBank corpus with word sense information and motivates the specific decisions taken with respect to the annotation schema and to the mapping between English and Bulgarian concepts, so that the synset hierarchy of the original WordNet can be preserved and reused.	2
14	Simova, I., Vasilev, D., Popov, A., Simov, K., & Osenova, P. (2014). Joint Ensemble Model for POS Tagging and Dependency Parsing. In Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-canonical Languages (pp. 15-25).	The work presented here combines several syntactic parsers and part-of-speech taggers in an ensemble system that is optimized to select the best decisions from the alternative modules, thus producing a solution that is globally optimal. The results from the separate systems are combined via a voting mechanism, which leads to improvements in terms of labeled and unlabeled attachment scores and part-of-speech tagging. It can be interpreted as a kind of motivation for doing multi-task learning – the system does not in fact optimize the separate modules based on joint-learning, but the improved results demonstrate that different systems learn different data dependencies, which can be explored in a multi-task setup.	3

Here I offer a list of citations from other researchers of the works described in the table.

• Simov, K., Popov, A., & Osenova, P. (2015). Improving Word Sense Disambiguation with Linguistic Knowledge from a Sense Annotated Treebank. In Proceedings of the International Conference Recent Advances in Natural Language Processing (pp. 596-603).

Cited by:

Hládek, Daniel, et al. "Survey of the word sense disambiguation and challenges for the Slovak language." Computational Intelligence and Informatics (CINTI), 2016 IEEE 17th International Symposium on. IEEE, 2016.

• Simov, K., Popov, A., & Osenova, P. (2016b). The Role of the WordNet Relations in the Knowledge-based Word Sense Disambiguation Task. In Proceedings of Eighth Global WordNet Conference (pp. 391–398).

Cited by:

Singh, Kuldeep, et al. "Why Reinvent the Wheel: Let's Build Question Answering Systems Together." Proceedings of the 2018 World Wide Web Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2018.

• Simov, K., Osenova, P., & Popov, A. (2016a). Towards Semantic-based Hybrid Machine Translation Between Bulgarian and English. In Proceedings of the 2nd Workshop on Semantics-Driven Machine Translation (SedMT 2016) (pp. 22–26).

Cited by:

Moussallem, Diego, Matthias Wauer, and Axel-Cyrille Ngonga Ngomo. "Machine Translation Using Semantic Web Technologies: A Survey." arXiv preprint arXiv:1711.09476 (2017).

• Simov, K., Osenova, P., & Popov, A. (2016b). Using Context Information for Knowledge-based Word Sense Disambiguation. In International Conference on Articial Intelligence: Methodology, Systems, and Applications (pp. 130-139).

Cited by:

Jelai, Lilyana, et al. "Textual Analysis by using Knowledge-based Word Sense Disambiguation Approach." Journal of Telecommunication, Electronic and Computer Engineering (JTEC) 9.3-3 (2017): 159-162.

Song, Xuebo, "Ontology-based Domain-specific Semantic Similarity Analysis and Applications" (2018). All Dissertations. 2105. https://tigerprints.clemson.edu/all_dissertations/2105

• Popov, A. (2016). Deep Learning Architecture for Part-of-speech Tagging with Word and Suffix Embeddings. In International Conference on Artificial Intelligence: Methodology, Systems, and Applications (pp. 68-77).

Cited by:

Bhargava, Rupal, Anushka Baoni and Yashvardhan Sharma. "Composite Sequential Modeling for Identifying Fake Reviews" Journal of Intelligent Systems, 0.0 (2018): -. Retrieved 12 Jul. 2018, from doi:10.1515/jisys-2017-0501

Farrah, Soufiane, Hanane El Manssouri, and Mohammed Ouzzif. "An hybrid approach to improve part of speech tagging system." Intelligent Systems and Computer Vision (ISCV), 2018 International Conference on. IEEE, 2018.

Wagner, Martin. Target Factors for Neural Machine Translation. Diss. Informatics Institute, 2017.

9.1.2 Approbation of the Results

Research Papers. The list of publications above describes works to which the author of the thesis has contributed. That work was done predominantly during the course of working on the PhD thesis, with two exceptions from 2014. All papers published as part of conference proceedings were presented at the conference by one of the authors. New work added by thesis to the body of published research is concentrated in chapters 7 (in terms of new models and experimental results) and 8 (where all presented material has not yet been published). Future work on new research papers on the same topics is also planned: developing further the results in the chapter on multi-task learning, producing better corpora for training distributed representations to be used by the *Architecture B* system in chapter 7, additional and better filtering techniques in the line of those presented in chapter 6.

Research Projects & Funding. Various parts of the work done in the context of this thesis have been funded by several research projects. Namely:

- The European Commission's FP7 project: QTLeap: Quality Translation by Deep Language Engineering Approaches (2014-2016)
- The European Commission's FP7 project: EUCases - EUropean and National CASE Law and Legislation Linked in Open Data Stack (2014-2015)
- Deep Models of Semantic Knowledge (DemoSem), funded by the Bulgarian National Science Fund in 2017–2019

Paper, Poster and Research Seminar Presentations. This is a list of presentations related to the thesis that the author has delivered in person:

1. 09/2015:

   Improving Word Sense Disambiguation with Linguistic Knowledge from a Sense Annotated Treebank. Conference presentation: Recent Advances in Natural Language Processing (RANLP). Hissar, Bulgaria 2015.

2. 06/2016:

   Towards Semantic-based Hybrid Machine Translation between Bulgarian and English. Semantics-Driven Machine Translation Workshop, collocated with the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL: HLT). San Diego, USA 2016.

3. 09/2016:

   Deep Learning Architecture for Part-of-Speech Tagging with Word and Suffix Embeddings. Conference presentation: Artificial Intelligence: Methodology, Systems, Applications (AIMSA). Varna, Bulgaria 2016.

4. 09/2016:

   Neural Network Language Models for Lexical Translation Suggestion. Workshop on Deep Language Processing for Quality Machine Translation, collocated with the AIMSA conference. Varna, Bulgaria 2016.

5. 09/2017:

   Word Sense Disambiguation with Recurrent Neural Networks. Poster: Student Workshop at RANLP. Varna, Bulgaria 2017.

6. 09/2017:

   Word Sense Disambiguation with Recurrent Neural Networks. Poster: Second International Summer School on Data Science 2017. Split, Croatia (2017).

7. 2017:

   Various presentations. DemoSem project internal research seminar at IICT, BAS. Sofia, Bulgaria.

9.1.3 Key Scientific and Applied Scientific Contributions

In this conclusion to the thesis it can be said that all tasks set out in the introductory chapter have been successfully accomplished. Many of the methods developed and results obtained can be further improved, but each task has been meaningfully engaged with and said engagement has led to concrete evaluation results. The tasks are once again listed and related to the chapters in the thesis in Table 9.2.

Work on these tasks has led to the following **scientific contributions**:

1. A detailed survey of the literature relevant to lexical modeling in NLP has been compiled, specifically aimed at bridging the gap between symbolic and probabilistic methods for encoding meaning. It includes a detailed overview of deep learning methods for lexical modeling, which constitute one of the most prominent foreseeable directions for future research.

2. It has been exhaustively demonstrated that modeling the lexicon through relational knowledge is a viable strategy. It has been shown that depending on the purpose of the modeling, various enrichments of the graph providing the relational knowledge will help in different ways. Paradigmatic relations tend to improve the modeling of lexical similarity, while syntagmatic relations tend to help for detecting relatedness of lexical items. Both kinds of relations are important in terms of the comprehensive modeling necessary to carry out complex lexical analyses such as word sense disambiguation.

Table 9.2: Successfully Conducted Tasks

Number	Task	Ref. to Thesis Chapter
1	Enhancing the WordNet semantic network for the purpose of knowledge-based word sense disambiguation.	5, 6
2	Designing and implementing a neural network architecture for sequence-to-sequence annotation.	4, 7
3	Applying the neural architecture to the task of part-of-speech tagging; experimenting with distributional morphosyntactic models as a source of features for learning.	4
4	Adapting the neural architecture for word sense disambiguation.	7
5	Deriving different distributional lexical models based on enrichments of the WordNet semantic network; testing the models on tasks such as knowledge-based and supervised WSD and similarity/relatedness calculation.	6, 7
6	Implementing neural systems for multi-task learning in order to test whether and to what extent different aspects of lexical knowledge interact with each other.	8

These observations have been motivated via numerous experiments that test the quality of different knowledge graphs with respect to various tasks: knowledge-based and supervised WSD, word similarity/relatedness calculation. A comparative analysis of the relation sets in WordNet, its glosses and such extracted from a syntactically annotated corpus has been carried out – demonstrating that different relations contribute differently to lexical modeling and motivating further research into how KG quality can be improved.

3. The thesis contributes strong evidence that encoding relational lexical knowledge in probabilistic distributed models is a powerful tool for lexical representation. It shows that improving the density and expressiveness of a KG can be successfully translated to vector space models, which can in turn be used as a representation source for supervised approaches to various tasks. Generating training data for such VSMs can be accomplished via simple algorithms for random walks on graph.

4. Several novel contributions in terms of representing the lexicon via vector space models are put forward. The first one is generating distributed

representations (embeddings) for a new kind of abstract constituent – *grammatical roles*, i.e. prototypical syntactic arguments of verbal predicates. The method allows for deriving theoretical knowledge from data; it can answer an important modeling question: what constitutes a good argument for a specific predicate (the approach can be extended to non-verbal predicates as well). Experimental results confirm its viability. The second contribution is a particular strategy for training distributed representations of lemmas and word senses/synsets in a shared space – by reusing the technique for the generation of pseudo-corpora. Experimental results show that the method is at least as successful as other popular approaches to the same problem. In addition to these two contributions, the thesis has also proposed a novel approach to encoding morphological information: the so-called *suffix embedding*, which is a simple method for capturing morphological dependencies. This kind of modeling has also produced positive results – on the task of POS tagging of Bulgarian text; to my knowledge, this is the first instance of applying a morphological analysis of this kind to the task at hand.

5. A new approach to supervised WSD has been proposed – one that employs recurrent neural networks, but instead of performing direct classification of senses at its output layer, learns to embed contextual representations of words in a mixed VSM that describes both words/lemmas and senses/synsets. The thesis has shown that improved mixed VSMs can make the models produced by the architecture competitive with top-performing systems, even though its main purpose is now WSD per se. A comparative analysis shows that classifier-based RNN architectures and context-embedding RNN architectures, such as those developed here, learn different representations and can be complementary to each other. The context-embedding approach has the advantage of always being able to produce a decision on its own, without having to rely on back-off strategies in the case of unknown words. In fact, without such a back-off strategy, the classifier-based system often is less accurate that the context-embedding one.

6. Multi-task learning is explored in the context of training jointly an RNN model on two different combinations of tasks: 1) sense classification and context embedding; 2) sense classification and POS tagging. In both cases the experimental results make a case for the strong interaction between the different aspects of lexical knowledge. This should serve as further motivation to explore this theoretical issue in closer detail.

There are number of **applied scientific contributions** as well:

1. Several RNN architectures for sequence-to-sequence tagging have been designed and implemented. More specifically, these solve the following tasks, which all engage with various aspects of lexical modeling: POS tagging, WSD and context representation. The results are close to the state of the art and indicate that further improvements and optimization might allow these systems to achieve it.

2. Multiple sets of new relations between word senses (synsets) have been generated, in the established format of WordNet. The new relation sets have been evaluated against different data sets, on two languages – Bulgarian and English. Significant gains in KBWSD have been achieved through the addition of the new resources to the KG. Different strategies for extracting relational knowledge from existing resources have been developed, including a filter-based approach that uses grammatical role embeddings in order to attempt an exhaustive automatic search for new relations in the graph.

3. Various vector space models have been trained and evaluated. Among them: VSMs for representing lemmas on the basis of semantic networks; mixed VSMs combining lemma, synset and in some cases grammatical role representations; word form and suffix embeddings for the Bulgarian language – the first such models trained for this language, as far as I am aware. The lemma embeddings achieve state-of-the-art results on word similarity/relatedness calculation, beating popular word embedding models; they also contribute significant features to the task of supervised WSD. The mixed lemma-synset models perform better than two popular such models, against which they are evaluated on the task of context embedding.

9.2 Outlook

The work described in the thesis has opened up many potential pathways for research into lexical modeling and associated fields. With the exception of POS tagging (where high accuracy scores have been achieved cross-linguistically and no great advancement can be expected at this point), all strands of research explored here can be followed further with an eye toward achieving still better experimental results. Moreover, advances in most of the tasks can be expected to have a positive

influence on the rest – as has been demonstrated, lexical representations encode multifarious kinds of information that serve as interfaces between different levels of linguistic analysis. For instance, further enriching the WordNet semantic network would potentially lead to better distributed representations of lemmas and synsets, which could in turn be used in conjunction with supervised WSD models. Below are listed some ideas for future development which have been considered by the author and colleagues.

Adapting information from additional resources to WordNet. There is an abundance of rich lexical resources that can be used to further enrich WordNet. As demonstrated in the thesis, this could lead to higher KBWSD scores and to VSMs that better capture lexical meaning. Some such resources that are worth considering and researching are: *VerbNet* (Schuler, 2005), *PropBank* (Kingsbury & Palmer, 2002), *FrameNet* (Baker et al., 1998), *BabelNet* (Navigli & Ponzetto, 2012). For instance, VerbNet contains a rich set of relations for most of the verb senses attested in English, including syntactic patterns, semantic valency frames and logic forms licensed by the verbs. The *SemLink* project[1] can be used to provide mappings between VerbNet and WordNet senses; the same resource also provides mappings to FrameNet, from where much more specific semantic relations can be extracted, and to PropBank, which is a real text corpus annotated with semantic arguments of verbs. Multilingual resources can also be explored for relevant relations.

Exhaustive inference of relations over WordNet using grammatical role embeddings. The experiments described in chapter 6 have shown that using distributed representations of prototypical syntactic arguments as filters for generating new relations can be a successful strategy. Much more work needs to be done, however. First of all, a more elaborate schema for the preprocessing of the training corpora would probably be very useful, since currently the encoding of the predicate arguments is very naive and does not discriminate very well between the different types of arguments; a different way to represent the corpus (e.g. via dependency paths) could lead to better distributed representations. Higher precision in parsing would also improve the resultant embeddings. Chapter 6 presented grammatical role embeddings only for a limited number of syntactic arguments, but in principle nothing stands in the way of generating such representations

[1] `ttps://verbs.colorado.edu/semlink/`

for all open word relations. Other types of relations (i.e. other than syntactic dependencies) could also serve as the basis for enriching WordNet (e.g. some kind of semantic relations, as long as a gold resource or an accurate enough parser is available).

Improving both architectures for WSD and the resulting models. Apart from further optimizations of the parameters of the networks, there are a number of possible options that present a somewhat natural path for their improvement. With regards to the classifier-based network (A), a CRF layer could be added on top, so that all choices would be globally optimal; another improvement would be to add an attention layer, which has been shown to boost results (Raganato, Bovi, & Navigli, 2017). Experimenting further with various VSMs is also an open task. With regards to the context-representation architecture (B), the most straightforward and promising pathway would be to continue improving the mixed VSM used for input features and gold label representations of synsets. Chapter 7 has shown that the VSM already performs better than other such mixed models. Improving it would depend on generating denser and better filtered semantic networks, as well as on post-processing of the generated artificial corpora, so that natural language text fragments can be intermixed with data from the knowledge graph. The last modification would improve the representation of function words and would also inject knowledge from real texts.

Further explorations of multi-task learning. Chapter 8 has demonstrated that multi-task learning can help improve accuracy on different kinds of tasks when those are combined. However, combinations that should be even more powerful from a theoretical point of view are available, such as WSD and syntactic parsing, WSD and semantic role labeling, etc. Even multi-modal learning could be explored, since information from other modalities (e.g. images) can provide rich information concerning lexical modeling.

Ensemble systems. The analysis of the results from the multi-task setup in chapter 8 has shown that *Architectures A and B* learn to make different choices in many cases. Figuring out how to combine the output of several such systems in a meaningful way could lead to going beyond the current state of the art.

Simplifying the WordNet dictionary. It has been pointed out numerous times in the research community (including in this thesis) that WordNet senses might be far too granular for some purposes. One such task that might benefit from coarser-grained sense distinctions is WSD, the state of the art on which is currently far behind a 90%-level accuracy that would make it viable for use in applications. This has been tried before (Navigli et al., 2007), but the task is inherently difficult in that a fine balance must be struck between making the lexicon coarse-grained enough for systems to achieve high accuracy scores and fine-grained enough for the annotations to actually be of any worth. Other difficulties exist as well. The OntoNotes corpus, for instance, is annotated with WordNet word senses that were clustered in such a way to ensure at least 90% inter-annotator agreement; however, clustering was done on the level of individual lemmas, not in terms of the synset hiearchy itself, which makes the WordNet semantic network more or less unusable in the ways described in this thesis. Clustering the synsets themselves would be a difficult task because solutions must be found to at least two difficult problems: how to identify useful clusters at the right level of granularity and how to transform the WordNet relations in a way that loses the least amount of information. VSMs like the ones described in the thesis could probably be used to solve the first problem; the second one remains an open task.

Declaration of Originality

Hereby, I declare that I have composed the presented thesis independently on my own and without any other resources than the ones indicated. All thoughts taken directly or indirectly from external sources are properly denoted as such.

This work has neither been previously submitted to another authority nor has it been published yet.

Place, Date, Signature

(*Name Family*)

Acknowledgments

Work on this dissertation has allowed me to focus on many exciting problems related to language analysis and requiring competence both in theoretical linguistics and in natural language processing. I want to thank first and foremost my scientific supervisor Kiril Simov for the many valuable ideas and for the guidance he has provided me with over the last four years. Without his patience, knowledge and keen mind, my own work would have been much poorer. It is thanks to him and to Petya Osenova that I have had the chance to work on various projects, attend valuable training programs and maintain some sort of continuity in my work reflecting my interests in lexical semantics and semantics in general. Several of the research papers that have provided the foundation for this thesis are collaborations with both of them and other colleagues to whom I also express my gratitude. The individual papers by myself also would not have been possible without support from our research group here at the Institute for Information and Communication Technologies in Sofia.

I am also grateful for the financial support without which this work would not have happened:

- The European Commission's FP7 project: QTLeap:Quality Translation by Deep Language Engineering Approaches. This project has funded some of the thesis-related work in the period 2014-2016.
- The European Commission's FP7 project: EUCases - EUropean and National CASE Law and Legislation Linked in Open Data Stack. The project has funded some of the thesis-related work in the period 2014-2015.
- Deep Models of Semantic Knowledge (DemoSem), funded by the Bulgarian National Science Fund. It has supported some of our group's work in 2017 and partly in 2018.

Bibliography

Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Paşca, M., & Soroa, A. (2009). A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (pp. 19–27).

Agirre, E., Barrena, A., & Soroa, A. (2015). Studying the Wikipedia Hyperlink Graph for Relatedness and Disambiguation. *arXiv preprint arXiv:1503.01655*.

Agirre, E., Bengoetxea, K., Gojenola, K., & Nivre, J. (2011). Improving Dependency Parsing with Semantic Classes. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2* (pp. 699–703).

Agirre, E., de Lacalle, O. L., & Soroa, A. (2014). Random Walks for Knowledge-based Word Sense Disambiguation. *Computational Linguistics*, *40*(1), 57–84.

Agirre, E., De Lacalle, O. L., Soroa, A., & Fakultatea, I. (2009). Knowledge-Based WSD and Specific Domains: Performing Better than Generic Supervised WSD. In *Ijcai* (pp. 1501–1506).

Agirre, E., & Rigau, G. (1996). Word Sense Disambiguation Using Conceptual Density. In *Proceedings of the 16th conference on Computational linguistics-Volume 1* (pp. 16–22).

Agirre, E., & Soroa, A. (2009). Personalizing PageRank for Word Sense Disambiguation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics* (pp. 33–41).

Alonso, H. M., & Plank, B. (2017). When is Multitask Learning Efective? Semantic Sequence Prediction Under Varying Data Conditions. In *15th Conference of the European Chapter of the Association for Computational Linguistics*.

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).

Banerjee, S., & Pedersen, T. (2003). Extended Gloss Overlaps as a Measure of Semantic Relatedness. In *Ijcai* (Vol. 3, pp. 805–810).

Baroni, M., Bernardini, S., Ferraresi, A., & Zanchetta, E. (2009). The WaCky Wide Web: a Collection of Very Large Linguistically PRocessed Web-crawled Corpora. *Language Resources and Evaluation*, *43*(3), 209–226.

Bentivogli, L., Forner, P., Magnini, B., & Pianta, E. (2004). Revising the WordNet Domains Hierarchy: Semantics, Coverage and Balancing. In *Proceedings of the Workshop on Multilingual Linguistic Ressources* (pp. 101–108).

Bergen, B. K. (2012). *Louder Than Words: The New Science of How the Mind Makes Meaning*. Basic Books (AZ).

Bond, F., Vossen, P., McCrae, J. P., & Fellbaum, C. (2016). CILI: The Collaborative Interlingual Index. In *Proceedings of the Global WordNet Conference* (Vol. 2016).

Borin, L., Forsberg, M., & Lönngren, L. (2013). SALDO: a Touch of Yin to WordNet's Yang. *Language Resources and Evaluation*, *47*(4), 1191–1211.

Brin, S., & Page, L. (1998). The Anatomy of a Large-scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems*, *30*(1-7), 107–117.

Brown, S. W., Dligach, D., & Palmer, M. (2014). VerbNet Class Assignment as a WSD Task. In *Computing Meaning* (pp. 203–216). Springer.

Cabezas, C., & Resnik, P. (2005). *Using WSD Techniques for Lexical Selection in Statistical Machine* (Tech. Rep.). Translation Technical report CS-TR-4736.

Camacho-Collados, J., Pilehvar, M. T., & Navigli, R. (2015). NASARI: a Novel Approach to a Semantically-aware Representation of Items. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 567–577).

Carpuat, M., & Wu, D. (2005). Word Sense Disambiguation vs. Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics* (pp. 387–394).

Carpuat, M., & Wu, D. (2007). Improving Statistical Machine Translation Using Word Sense Disambiguation. In *EMNLP-CoNLL* (Vol. 7, pp. 61–72).

Chan, Y. S., Ng, H. T., & Chiang, D. (2007). Word Sense Disambiguation Improves Statistical Machine Translation. In *Annual Meeting-Association for Computational Linguistics* (Vol. 45, p. 33).

Chen, X., Liu, Z., & Sun, M. (2014). A Unified Model for Word Sense Representation and Disambiguation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1025–1035).

Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the Properties of Neural Machine Translation: Encoder-decoder Approaches. *arXiv preprint arXiv:1409.1259*.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv preprint arXiv:1412.3555*.

Ciaramita, M., & Altun, Y. (2006). Broad-coverage Sense Disambiguation and Information Extraction with a Supersense Sequence Tagger. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing* (pp. 594–602).

Collins, M. (2003). Head-driven Statistical Models for Natural Language Parsing. *Computational linguistics*, *29*(4), 589–637.

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).

Copestake, A., Flickinger, D., Pollard, C., & Sag, I. A. (2005). Minimal Recursion Semantics: An Introduction. *Research on Language and Computation*, *3*(2-3), 281–332.

Dalrymple, M. (2001). *Lexical Functional Grammar*. Brill.

Edmonds, P., & Cotton, S. (2001). SENSEVAL-2: Overview. In *The Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems* (pp. 1–5).

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., & Lin, C.-J. (2008). LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research*, *9*(Aug), 1871–1874.

Faruqui, M., Dodge, J., Jauhar, S. K., Dyer, C., Hovy, E., & Smith, N. A. (2014). Retrofitting Word Vectors to Semantic Lexicons. *arXiv preprint arXiv:1411.4166*.

Fellbaum, Christiane. (1998). *Wordnet*. Wiley Online Library.

Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., & Ruppin, E. (2001). Placing Search in Context: The Concept Revisited. In *Proceedings of the 10th International Conference on World Wide Web* (pp. 406–414).

Galley, M., & McKeown, K. (2003). Improving Word Sense Disambiguation in Lexical Chaining. In *IJCAI* (Vol. 3, pp. 1486–1488).

Giuglea, A.-M., & Moschitti, A. (2006). Semantic Role Labeling via FrameNet, VerbNet and PropBank. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics* (pp. 929–936).

Goikoetxea, J., Agirre, E., & Soroa, A. (2016). Single or Multiple? Combining Word Representations Independently Learned from Text and WordNet. In *AAAI* (pp. 2608–2614).

Goikoetxea, J., Soroa, A., Agirre, E., & Donostia, B. C. (2015). Random Walks and Neural Network Language Models on Knowledge Bases. In *HLT-NAACL* (pp. 1434–1439).

Graves, A. (2012). Supervised Sequence Labelling. In *Supervised Sequence Labelling with Recurrent Neural Networks* (pp. 5–13). Springer.

Graves, A., Mohamed, A.-r., & Hinton, G. (2013). Speech Recognition with Deep Recurrent Neural Networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE international conference on* (pp. 6645–6649).

Halliday, M. A., & Hasan, R. (1976). Cohesion in English. *English, Longman, London*.

Hendrickx, I., Kim, S. N., Kozareva, Z., Nakov, P., Ó Séaghdha, D., Padó, S., ... Szpakowicz, S. (2009). Semeval-2010 Task 8: Multi-way Classification of Semantic Relations Between Pairs of Nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions* (pp. 94–99).

Hill, F., Reichart, R., & Korhonen, A. (2015). Simlex-999: Evaluating Semantic Models with (Genuine) Similarity Estimation. *Computational Linguistics*, *41*(4), 665–695.

Huang, Z., Xu, W., & Yu, K. (2015). Bidirectional LSTM-CRF Models for Sequence Tagging. *arXiv preprint arXiv:1508.01991*.

Iacobacci, I., Pilehvar, M. T., & Navigli, R. (2015). SensEmbed: Learning Sense Embeddings for Word and Relational Similarity. In *ACL (1)* (pp. 95–105).

Iacobacci, I., Pilehvar, M. T., & Navigli, R. (2016). Embeddings for Word Sense Disambiguation: An Evaluation Study. In *ACL (1)*.

Jackendoff, R. (1992). *Semantic Structures* (Vol. 18). MIT press.

Johansson, R., & Pina, L. N. (2015). Embedding a Semantic Network in a Word Space. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 1428–1433).

Kågebäck, M., & Salomonsson, H. (2016). Word Sense Disambiguation Using a Bidirectional LSTM. *arXiv preprint arXiv:1606.03568*.

Kenter, T., Borisov, A., & de Rijke, M. (2016). Siamese CBOW: Optimizing Word Embeddings for Sentence Representations. *arXiv preprint arXiv:1606.04640*.

Kilgarriff, A. (2001). English Lexical Sample Task Description. In *The Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems* (pp. 17–20).

Kingsbury, P., & Palmer, M. (2002). From TreeBank to PropBank. In *LREC* (pp. 1989–1993).

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Skip-thought Vectors. In *Advances in Neural Information Processing Systems* (pp. 3294–3302).

Kučera, H., & Francis, W. N. (1967). *Computational Analysis of Present-day American English.* Dartmouth Publishing Group.

Kucera, H., & Francis, W. N. (1982). *Frequency Analysis of English Usage: Lexicon and Grammar.* Boston: Houghton Mifflin.

Le, M., Postma, M., & Urbani, J. (2017). Word Sense Disambiguation with LSTM: Do We Really Need 100 Billion Words? *arXiv preprint arXiv:1712.03376*.

Le, Q., & Mikolov, T. (2014). Distributed Representations of Sentences and Documents. In *International Conference on Machine Learning* (pp. 1188–1196).

Lesk, M. (1986). Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation* (pp. 24–26).

Levin, B. (1993). *English Verb Classes and Alternations: A Preliminary Investigation.* University of Chicago press.

Levy, O., & Goldberg, Y. (2014). Dependency-based Word Embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (Vol. 2, pp. 302–308).

Ling, W., Luís, T., Marujo, L., Astudillo, R. F., Amir, S., Dyer, C., ... Trancoso, I. (2015). Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation. *arXiv preprint arXiv:1508.02096*.

MacKinlay, A., Dridan, R., McCarthy, D., & Baldwin, T. (2012). The Effects of Semantic Annotations on Precision Parse Ranking. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation* (pp. 228–236).

Mancini, M., Camacho-Collados, J., Iacobacci, I., & Navigli, R. (2016). Embedding Words and Senses Together via Joint Knowledge-Enhanced Training. *arXiv preprint arXiv:1612.02703*.

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.

Melamud, O., Goldberger, J., & Dagan, I. (2016). context2vec: Learning Generic Context Embedding with Bidirectional LSTM. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning* (pp. 51–61).

Mihalcea, R. (2005). Unsupervised Large-vocabulary Word Sense Disambiguation with Graph-based Algorithms for Sequence Data Labeling. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing* (pp. 411–418).

Mihalcea, R., Chklovski, T., & Kilgarriff, A. (2004). The Senseval-3 English Lexical Sample Task. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text.*

Mihalcea, R., & Moldovan, D. I. (2001). eXtended WordNet: Progress Report. In *in Proceedings of NAACL Workshop on WordNet and Other Lexical Resources.*

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Deoras, A., Kombrink, S., Burget, L., & Černockỳ, J. (2011). Empirical Evaluation and Combination of Advanced Language Modeling Techniques. In *Twelfth Annual Conference of the International Speech Communication Association.*

Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., & Khudanpur, S. (2010). Recurrent Neural Network Based Language Model. In *Interspeech* (Vol. 2, p. 3).

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed Representations of Words and Phrases and Their Compositionality. In *Advances in Neural Information Processing Systems* (pp. 3111–3119).

Miller, G. A., Leacock, C., Tengi, R., & Bunker, R. T. (1993). A Semantic Concordance. In *Proceedings of the Workshop on Human Language Technology* (pp. 303–308).

Minsky, M., & Papert, S. A. (2017). *Perceptrons: an Introduction to Computational Geometry.* MIT press.

Moldovan, D. I., & Rus, V. (2001). Logic Form Transformation of WordNet and its Applicability to Question Answering. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics* (pp. 402–409).

Moro, A., Cecconi, F., & Navigli, R. (2014). Multilingual Word Sense Disambiguation and Entity Linking for Everybody. In *Proceedings of the 2014 International Conference on Posters & Demonstrations Track-Volume 1272* (pp. 25–28).

Moro, A., & Navigli, R. (2015). Semeval-2015 task 13: Multilingual All-words Sense Disambiguation and Entity Linking. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* (pp. 288–297).

Moro, A., Raganato, A., & Navigli, R. (2014). Entity Linking Meets Word Sense Disambiguation: a Unified Approach. *Transactions of the Association for Computational Linguistics*, *2*, 231–244.

Navigli, R. (2009). Word Sense Disambiguation: A survey. *ACM Computing Surveys (CSUR)*, *41*(2), 10.

Navigli, R., Jurgens, D., & Vannella, D. (2013). Semeval-2013 Task 12: Multilingual Word Sense Disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)* (Vol. 2, pp. 222–231).

Navigli, R., & Lapata, M. (2007). Graph Connectivity Measures for Unsupervised Word Sense Disambiguation. In *IJCAI* (pp. 1683–1688).

Navigli, R., & Lapata, M. (2010). An Experimental Study of Graph Connectivity for Unsupervised Word Sense Disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(4), 678–692.

Navigli, R., Litkowski, K. C., & Hargraves, O. (2007). Semeval-2007 Task 07: Coarse-grained ENglish All-words Task. In *Proceedings of the 4th International Workshop on Semantic Evaluations* (pp. 30–35).

Navigli, R., & Ponzetto, S. P. (2012). BabelNet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network. *Artificial Intelligence*, *193*, 217–250.

Navigli, R., & Velardi, P. (2005). Structural Semantic Interconnections: a Knowledge-based Approach to Word Sense Disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(7), 1075–1086.

Oele, D., & van Noord, G. (2018). Simple Embedding-Based Word Sense Disambiguation..

Papandrea, S., Raganato, A., & Bovi, C. D. (2017). SUPWSD: A Flexible Toolkit for Supervised Word Sense Disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (pp. 103–108).

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543).

Petrov, S., Das, D., & McDonald, R. (2011). A Universal Part-of-speech Tagset. *arXiv preprint arXiv:1104.2086*.

Plank, B., Søgaard, A., & Goldberg, Y. (2016). Multilingual Part-of-speech Tagging with Bidirectional Long Short-term Memory Models and Auxiliary Loss. *arXiv preprint arXiv:1604.05529*.

Pollard, C., & Sag, I. A. (1994). *Head-driven Phrase Structure Grammar*. University of Chicago Press.

Popov, A. (2016a). Deep Learning Architecture for Part-of-Speech Tagging with Word and Suffix Embeddings. In *International Conference on Artificial Intelligence: Methodology, Systems, and Applications* (pp. 68–77).

Popov, A. (2016b). Neural Network Language Models–an Overview. In *The Workshop on Deep Language Processing for Quality Machine Translation (DeepLP4QMT)* (p. 20-26).

Popov, A. (2017). Word Sense Disambiguation with Recurrent Neural Networks. In *Proceedings of the Student Research Workshop Associated with RANLP 2017* (pp. 25–34).

Popov, A. (2018). Neural Network Models for Word Sense Disambiguation: an Overview. *Cybernetics and Information Technologies*, *18*(1), 139–151.

Popov, A., Kancheva, S., Manova, S., Radev, I., Simov, K., & Osenova, P. (2014). The Sense Annotation of BulTreeBank. *Proceedings of TLT13*, 127–136.

Pradhan, S. S., Loper, E., Dligach, D., & Palmer, M. (2007). SemEval-2007 task 17: English Lexical Sample, SRL and All Words. In *Proceedings of the 4th International Workshop on Semantic Evaluations* (pp. 87–92).

Pustejovsky, J. (1991). The generative lexicon. *Computational Linguistics*, *17*(4), 409–441.

Pustejovsky, J. (1995). *The Generative Lexicon*. MIT Press.

Rada, R., Mili, H., Bicknell, E., & Blettner, M. (1989). Development and Application of a Metric on Semantic Nets. *IEEE Transactions on Systems, Man, and Cybernetics*, *19*(1), 17–30.

Raganato, A., Bovi, C. D., & Navigli, R. (2017). Neural Sequence Learning Models for Word Sense Disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (pp. 1156–1167).

Raganato, A., Camacho-Collados, J., & Navigli, R. (2017). Word Sense Disambiguation: A Unified Evaluation Framework and Empirical Comparison. In *Proc. of EACL* (pp. 99–110).

Rink, B., & Harabagiu, S. (2010). Utd: Classifying Semantic Relations by Combining Lexical and Semantic Resources. In *Proceedings of the 5th International Workshop on Semantic Evaluation* (pp. 256–259).

Ristoski, P., & Paulheim, H. (2016). Rdf2vec: Rdf Graph Embeddings for Data Mining. In *International Semantic Web Conference* (pp. 498–514).

Rothe, S., & Schütze, H. (2015). Autoextend: Extending Word Embeddings to Embeddings for Synsets and Lexemes. *arXiv preprint arXiv:1507.01127*.

Sanderson, M. (1994). Word Sense Disambiguation and Information Retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 142–151).

Schuler, K. K. (2005). VerbNet: A Broad-coverage, Comprehensive Verb Lexicon.

Schuster, M., & Paliwal, K. K. (1997). Bidirectional Recurrent Neural Networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681.

Schütze, H., & Pedersen, J. O. (1995). Information Retrieval Based on Word Senses.

Simov, K., & Osenova, P. (2001). A Hybrid System for Morphosyntactic Disambiguation in Bulgarian. In *Proceedings of the EuroConference on Recent Advances in Natural Language Processing* (pp. 5–7).

Simov, K., & Osenova, P. (2004). *BTB-TR04: BulTreeBank Morphosyntactic Annotation of Bulgarian Texts* (Tech. Rep.). Technical Report BTB-TR04, Bulgarian Academy of Sciences.

Simov, K., Osenova, P., & Popov, A. (2016a). Towards Semantic-based Hybrid Machine Translation Between Bulgarian and English. In *Proceedings of the 2nd Workshop on Semantics-Driven Machine Translation (SedMT 2016)* (pp. 22–26).

Simov, K., Osenova, P., & Popov, A. (2016b). Using Context Information for Knowledge-based Word Sense Disambiguation. In *International Conference on Artificial Intelligence: Methodology, Systems, and Applications* (pp. 130–139).

Simov, K., Osenova, P., & Popov, A. (2017). Comparison of Word Embeddings from Different Knowledge Graphs. In *International Conference on Language, Data and Knowledge* (pp. 213–221).

Simov, K., Osenova, P., & Slavcheva, M. (2004). *BulTreeBank Morphosyntactic Tagset* (Tech. Rep.). Technical Report BTB-TR03, BulTreeBank Project.

Simov, K., Osenova, P., Slavcheva, M., Kolkovska, S., Balabanova, E., Doikoff, D., ... Kouylekov, M. (2002). Building a Linguistically Interpreted Corpus of Bulgarian: the BulTreeBank. In *LREC.*

Simov, K., Popov, A., & Osenova, P. (2015). Improving Word Sense Disambiguation With Linguistic Knowledge from a Sense Annotated Treebank. In *Proceedings of the International Conference Recent Advances in Natural Language Processing* (pp. 596–603).

Simov, K., Popov, A., & Osenova, P. (2016a). Knowledge Graph Extension for Word Sense Annotation. In *Innovative Approaches and Solutions in Advanced Intelligent Systems* (pp. 151–166). Springer.

Simov, K., Popov, A., & Osenova, P. (2016b). The Role of the WordNet Relations in the Knowledge-based Word Sense Disambiguation Task. In *Proceedings of Eighth Global WordNet Conference* (pp. 391–398).

Simov, K., Popov, A., Simova, I., & Osenova, P. (2018). Grammatical Role Embeddings for Enhancements of Relation Density in the Princeton WordNet. In *Proceedings of the 9th Global Wordnet Conference.*

Simov, K., Popov, A., Zlatkov, L., & Kotuzov, N. (2016). Transfer of Deep Linguistic Knowledge in a Hybrid Machine Translation System. In *The Workshop on Deep Language Processing for Quality Machine Translation (DeepLP4QMT)* (p. 27-33).

Simova, I., Vasilev, D., Popov, A., Simov, K., & Osenova, P. (2014). Joint Ensemble Model for POS Tagging and Dependency Parsing. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages* (pp. 15–25).

Snyder, B., & Palmer, M. (2004). The English All-words Task. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text.*

Socher, R., Lin, C. C., Manning, C., & Ng, A. Y. (2011). Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)* (pp. 129–136).

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*(1), 1929–1958.

Stokoe, C., Oakes, M. P., & Tait, J. (2003). Word Sense Disambiguation in Information Retrieval Revisited. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval* (pp. 159–166).

Sundermeyer, M., Ney, H., & Schlüter, R. (2015). From Feedforward to Recurrent LSTM Neural Networks for Language Modeling. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, *23*(3), 517–529.

Sussna, M. (1993). Word Sense Disambiguation for Free-text Indexing Using a Massive Semantic Network. In *Proceedings of the Second International Conference on Information and Knowledge Management* (pp. 67–74).

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).

Taghipour, K., & Ng, H. T. (2015). Semi-Supervised Word Sense Disambiguation Using Word Embeddings in General and Specific Domains. In *HLT-NAACL* (pp. 314–323).

Tong, H., Faloutsos, C., & Pan, J.-Y. (2006). Fast Random Walk with Restart and its Applications.

Vickrey, D., Biewald, L., Teyssier, M., & Koller, D. (2005). Word-sense Disambiguation for Machine Translation. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing* (pp. 771–778).

Wang, P., Qian, Y., Soong, F. K., He, L., & Zhao, H. (2015a). Part-of-speech Tagging with Bidirectional Long Short-term Memory Recurrent Neural Network. *arXiv preprint arXiv:1510.06168*.

Wang, P., Qian, Y., Soong, F. K., He, L., & Zhao, H. (2015b). A Unified Tagging Solution: Bidirectional LSTM Recurrent Neural Network With Word Embedding. *arXiv preprint arXiv:1511.00215*.

Weischedel, R., Palmer, M., Marcus, M., Hovy, E., Pradhan, S., Ramshaw, L., ... others (2013). Ontonotes Release 5.0 LDC2013T19. *Linguistic Data Consortium, Philadelphia, PA*.

Yin, W., Kann, K., Yu, M., & Schütze, H. (2017). Comparative Study of CNN and RNN for Natural Language Processing. *arXiv preprint arXiv:1702.01923*.

Yuan, D., Richardson, J., Doherty, R., Evans, C., & Altendorf, E. (2016). Semi-supervised Word Sense Disambiguation with Neural Models. *arXiv preprint arXiv:1603.07012*.

Zapirain, B., Agirre, E., Marquez, L., & Surdeanu, M. (2013). Selectional Preferences for Semantic Role Classification. *Computational Linguistics*, *39*(3), 631–663.

Zhong, Z., & Ng, H. T. (2010). It Makes Sense: A Wide-coverage Word Sense Disambiguation System for Free text. In *Proceedings of the ACL 2010 System Demonstrations* (pp. 78–83).

Appendix A

List of Tables

Appendix B

List of Figures

Appendix C

List of Abbreviations

Bi-LSTM	Bidirectional Long Short-term Memory
IMS	It Makes Sense
IR	Information Retrieval
KB	Knowledge Base
KBWSD	Knowledge-based Word Sense Disambiguation
KG	Knowledge Graph
LSTM	Long Short-term Memory
LM	Language Model
ML	Machine Learning
MT	Machine Translation
NER	Named Entity Recognition
NLP	Natural Language Processing
NN	Neural Network
POS	Part-of-speech
RNN	Recurrent Neural Network
SVM	Support Vector Machine
VSM	Vector Space Model
WN	WordNet
WSD	Word Sense Disambiguation