Lexical Modeling for Natural Language Processing

Alexander Popov
Institute of Information and Communication Technologies
Bulgarian Academy of Sciences, Bulgaria
alex.popov@bultreebank.org

Abstract

This paper describes a multi-component research project on the computational lexicon, the results of which will be used and built upon in work within the CLARIN infrastructure to be developed by the Bulgarian national consortium. Princeton WordNet is used as the primary lexicographic resource for producing machine-oriented models of meaning. Its dictionary and semantic network are used to build knowledge graphs, which are then enriched with additional semantic and syntactic relations extracted from various other sources. Experimental results demonstrate that this enrichment leads to more accurate lexical analysis. The same graph models are used to create distributed semantic models (or ”embeddings”), which perform very competitively on standard word similarity and relatedness tasks. The paper discusses how such vector models of the lexicon can be used as input features to neural network systems for word sense disambiguation. Several neural architectures are discussed, including two multi-task architectures, which are trained to reflect more accurately the polyvalent nature of lexical items. Thus, the paper provides a faceted view of the computational lexicon, in which separate aspects of it are modeled in different ways, relying on different theoretical and data sources, and are used to different purposes.

1 Introduction

In many theories, both within theoretical and computational linguistics, the lexicon plays the role of bridging language structures. Projects like VerbNet (Schuler, 2005), which extends Beth Levin's work on verb classes (1993), aim to bring together various levels of linguistic analysis – syntactic, semantic, predicate logic – and make them cohere via the lexical interface. Other projects like SemLink (Palmer, 2009) bind together even more heterogeneous data: VerbNet, FrameNet, PropBank, and recently the WordNet sense groupings used in the OntoNotes corpus (Bonial et. al., 2013). The Predicate Matrix project in turn extends SemLink to obtain an even wider coverage (De Lacalle et. al., 2014). Such a model of the lexicon moves toward a more realistic representation that reflects the interdependence of linguistic structures.

Combining heterogeneous data in computational lexicons is a challenge that has been taken up with renewed vigour in recent years, due to the increased popularity of *vector space models* (VSMs), also known in the literature as *embeddings*. New, more efficient approaches to embedding (Mikolov et. al., 2013b; Pennington et. al., 2014) allow for the learning of latent information from huge volumes of text and encoding that in real-valued vectors of varying dimensionality. In this way semantico-syntactic features can be extracted and compressed in powerful and compact models that are more easily interpretable by machine learning (ML) systems. This excitement around embedding methods has led to growing research in *sense embedding*, i.e. methods to derive embeddings for other linguistic items, namely word senses. Since training data annotated with word senses is much more difficult to obtain, various approaches have been tried, with most of them hingeing on somehow translating the structure of lexical resources (such as the knowlege graph of WordNet) into instructions for training (Faruqui et. al., 2014; Johansson and Pina, 2015; Rothe and Schütze, 2015; Goikoetxea et. al., 2015).

The present paper describes the first phase of a hybrid approach for the construction of such multi-aspectual representations of the lexicon that are suitable for use by automatic systems. The representations take different forms: knowledge graphs, vector space models, neural networks trained on lexical analysis tasks. This line of research assumes WordNet (Fellbaum, 2010) as its main knowledge resource and tests a number of hypotheses about how this computational lexicon can be enriched and adapted to various natural language processing (NLP) tasks, such as word sense disambiguation (WSD), word similarity/relatedness, context representation, part-of-speech (POS) tagging. The enriched resources and ML models will be integrated in and developed further within the CLaDA-BG project[1], in order to reflect a rich multi-level view of the lexicon. Though this paper focuses on English data, Bulgarian models will be produced as well.

This paper presents current results and perspectives. More specifically, it describes:

- the effects of knowledge graph enrichment on knowledge-based WSD accuracy;
- graph embedding on enriched knowledge graphs;
- using graph embedding models for supervised WSD;
- benefits from multi-task supervised lexical learning with recurrent neural networks.

2 Related Work

Knowledge-based WSD Navigli (2009) provides a good overview of knowledge-based WSD (KBWSD). The current work deals exclusively with KBWSD via graphs, building on Agirre and Soroa (2009) and using the UKB software[2] for PageRank-based WSD. The latter work uses WordNet's original semantic network, relations between the manual annotations of the glosses, and the automatic annotations of glosses from eXtended WordNet (Harabagiu and Moldovan, 2000) to construct its knowledge graph (KG). Agirre et. al. (2018) describe new settings for that software that make the approach state-of-the-art within KBWSD and very competitive even against supervised systems.

Lexical learning with RNNs Various aspects of lexical learning have been tackled successfully with deep neural networks. Recurrent neural networks (RNNs) in particular are a powerful tool for handling sequences of language tokens. With respect to WSD, Kågebäck and Salomonsson (2016) have obtained state-of-the-art results on the *lexical sample task*, using bidirectional Long Short-term Memory (Bi-LSTM) networks, and Raganato et. al. (2017b) have done so on the *all-words task*. With respect to POS tagging, various papers have demonstrated the viability or RNNs (Wang et. al., 2015a; Wang et. al., 2015b; Huang et. al., 2015). Context embedding has become a popular method in NLP, especially within the deep learning paradigm. Kiros et. al. (2015) describe a model for learning a generic sentence encoder that provides distributed representations which can be used in different settings. Melamud et. al. (2016) present another neural architecture for context representation – called *context2vec* – which can be used for WSD.

Multi-task learning Raganato et. al. (2017b) combine WSD, POS tagging and coarse-grained semantic labeling (CGSL) in an RNN setup. The empirical comparison shows that with a number of data sets training on the coarse-grained task does help in improving the accuracy of WSD. The addition of POS tagging to the combination WSD+CGSL is not universally beneficial – it increases the accuracy of WSD on several data sets, but brings it down on others. Plank et. al. (2016) present a Bi-LSTM tagger tested on data for 22 languages; the addition of a frequency classification auxiliary task to the RNN is shown to improve results on many of the languages. Alonso and Plank (2016) 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). Nguyen et. al. (2017) present a state-of-the-art joint Bi-LSTM model for POS tagging and graph-based dependency parsing. Ruder (2017) is a good comprehensive overview of various multi-task learning setups, while

[1]National interdisciplinary Research e-Infrastructure for Resources and Technologies in favor of the Bulgarian Language and Cultural Heritage, as part of the EU infrastructures CLARIN and DARIAH. Grant number D01-164/28.08.2018 by Bulgarian Ministry of Education and Science.

[2]`http://ixa2.si.ehu.es/ukb/`

Bingel and Søgaard (2017) evaluate all possible combinations of two NLP tasks out of a pool of ten tasks.

Vector space models for word senses (synsets) One approach for training sense embeddings is through *retrofitting methods*, i.e. adapting an already existing resource to reflect the structure of a knowledge base, typically WordNet (Faruqui et. al., 2014; Johansson and Pina, 2015; Rothe and Schütze, 2015). Camacho-Collados et. al. (2015) put forward NASARI – a system for automatically constructing vector representations for WordNet synsets and Wikipedia pages. Another way to learn distributed representations of senses is to automatically sense-annotate huge amounts of natural text and then train a neural network language model on the processed data. Iacobacci et. al. (2015) use the BabelNet sense inventory and the Babelfy knowledge-based WSD tool to tag a dump of Wikipedia on which to train the VSMs. Mancini et. al. (2016) propose an interesting solution called *SW2V*; they use a *shallow word-sense connectivity algorithm* in order to annotate the open class words of a large corpus with potential word senses, then train a modified CBOW architecture (Mikolov et. al., 2013b) on word and sense prediction simultaneously. A somewhat different way of creating distributed representations of concepts is via generating artificial sequences from knowledge graphs, on which neural network language models can be trained (Goikoetxea et. al., 2015). This is the approach adopted in this work. It allows to easily modify the training data and to produce hybrid VSMs (e.g. combinations of words and word senses). The article provides empirical evidence that it is very competitive compared to the other methods introduced above.

3 Graph Enrichment for Knowledge-based Word Sense Disambiguation

The experiments reported in this section present a number of strategies for obtaining a fuller lexical representation of words and senses, with WordNet (WN) as a starting point. The main idea is to add new relations to the original WN semantic network and to evaluate the effect of this modification on a KBWSD algorithm. The section first describes several strategies for adding new relations and then provides results on English sense-tagged data.

Preliminary experiments on Bulgarian data have shown that using a syntactically and semantically annotated corpus in conjunction with the hypernym-hyponym hierarchy from WN to generate new syntagmatic relations between concepts can significantly improve KBWSD accuracy (Simov et. al., 2015). Additional strategies have been explored for the extraction of new relations from sense-annotated data for English. The various approaches, including the baseline relations from the original WN and eXtended WordNet (XWN) networks, are summarized and abbreviated below:

- The original WN semantic network, comprising of lexical semantic relations, such as hypernymy, antonymy, meronymy, etc. – **WN**
- The additional relations between open class words extracted from the manually annotated WN glosses – **WNG**
- Logic form representations and parse trees from eXtended WordNet (XWN). XWN contains first-order logic representations of the glosses, from which ”event” relations can be extracted – **WNGL**
- Context representation of XWN annotations. Word order is used to connect word sense annotations in the glosses. Each sense annotation is indirectly connected to the one preceding it via an artificial node that indicates contextual dependency. The artificial nodes too are added to the graph – **WN30glCon**
- Syntactic relations from the SemCor corpus (Miller et. al., 1993). The sense-annotated corpus is analyzed with a dependency parser. The sense annotations are attached to the nodes in the dependency trees, so that the full syntactic information about context usage is inserted in the graph (including functional words). Sentences are also connected to the ones preceding them via an artificial node that connects their root nodes – **GraphRelSC**

For a more detailed description of the relation sets, see Simov et. al. (2016).

Table 1 shows experimental results with a selection of the best KG combinations. Two data sets are used for evaluation. The first one is WSDEval – a concatenation of the Senseval-2, Seneval-3, SemEval-07, SemEval-13, SemEval-15 data sets, as described in Raganato et. al. (2017a)[3]. The second one is a subset of SemCor (49 files in total) which was not used for the extraction of new relations. The annotation was performed with the UKB system, using the settings outlined in Agirre et. al. (2018). The addition of new relations improves accuracy scores in all cases of evaluation on the SemCor data, while it actually hurts scores on the WSDEval data set in two of the three setups with enriched graphs. However, the combination of the baseline relations and the syntactic enrichment extracted from SemCor provides a big boost to the KBWSD algorithm[4]. The system is thus able not just to achieve the best reported results for KBWSD (at least in Raganato et. al. (2017a)), but to come close even to the top supervised WSD models.

KG	$WSDEval$	$SemCor$
WN + WNG	67.30%	74.2%
WN + WNG + WNGL	66.9%	74.9%
WN + WNG + WN30glCon	67.1%	**75.1%**
WN + WNG + GraphRelSC	**68.2%**	**75.1%**

Table 1: KBWSD accuracy on WSDEval and SemCor.

4 Graph Embedding

Building on the observation that enriching the KG does indeed help with lexical analysis tasks such as KBWSD, the line of research presented in this section attempts to translate this type of structured information (KG) into a numerical representation (vector space model) and evaluate the latter on standardly used tasks. To do that, the method in Goikoetxea et. al. (2015) is used. It is outlined below, in combination with the preliminary step of graph enrichment:

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 a lemma from the corresponding synset. When training word or lemma embeddings, the probability for outputting a lemma is set to 1. Each random walk constitutes one training sentence.
3. The Word2Vec tool[5] is used to train a VSM on the artificially produced corpus, in particular the Skip-Gram architecture. The following parametrization is employed: 7 iterations over the data; number of negative examples set to 5; frequency cut sampling set to 7; context windows of size 5 or 15.
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 (Agirre et. al. (2009) describe the two subsets of the WordSim-353 data set) and SimLex-999 (Hill et. al., 2015) data sets are used for the evaluation. 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.

Table 2 shows the results with a selection of the best performing VSMs on the three evaluation data sets. The first three results are obtained with the baseline models: the **GoogleNews** model trained on a large corpora of natural language text; vectors trained on contexts generated from dependency parses

[3]This is part of the Unified Evaluation Framework (UEF) resource, assessible at `http://lcl.uniroma1.it/wsdeval/`

[4]When used in combination, WNGL, WN30glCon, GraphRelSC do not yield results that are better than the baseline. This indicates that with such parametrization of the algorithm two of the three new relation sets actually introduce noise rather than useful knowledge.

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

of natural language text (**Dependency**); the WordNet-generated pseudo-corpus (**WN + WNG**), which is produced from the original semantic network and the gloss annotations. The first two new models (**WN + WNG + HypInf** *C5/C15*) add the transitive closure over the hypernym-hyponym relations in WN. This means that the direct taxonomic relations are made explicit between all elements in a hypernymy chain: e.g. if ”surgeon” is marked as a hyponym of ”doctor” and ”doctor” is marked as a hyponym of ”medical professional”, an explicit link is added between ”surgeon” and ”medical professional”; this is done for all implicit taxonomic relations of this nature. The table shows that this addition significantly improves the performance on the *SimLex999* data set, even though it reduces the performance on the rest of the data. The second group of models based on the extended graphs (**WN + WNglConOne** *C5/C15*) combines the original WN semantic network with the **WNglConOne** set of relations introduced in the previous section. Already these combinations achieve scores that are better than or at worst on par with all three baselines (with the VSM trained on a context window of 15 words being significantly better on two out of three data sets). The third group of models (**WN + WNG + WNGL + GrRelSC** + *C5/C15*) is based on data generated via the combination of WN, the gloss annotations, the relations extracted from the logic forms in XWN and of the syntactic relations from SemCor. These models achieve the best results on the *WordSim353* similarity data set.

Vector Space Model	*WordSim353 Similarity*	*WordSim353 Relatedness*	*SimLex999*
GoogleNews(Mikolov et. al., 2013a)	0.77145	0.61988	0.44196
Dependency(Levy et. al., 2014)	0.76699	0.46764	0.44730
WN + WNG(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 2: 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 a source, are marked in bold. The final two experiments 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.

It is evident that different types of new relations contribute to performance differently on the different data sets. For instance, hypernymy-derived relations lead to improvements on the similarity rather than on the relatedness task, possibly because paradigmatic relations in general are more informative with regards to semantic similarity. The experiments also show that combining different kinds of relation sets is beneficial, i.e. there is a significant degree of complementarity between them. This result provides justification for integrating the various sources of information in a common lexicon model. The last two experiments take one of the best models (**WN + WNglConOne** *C15*) and concatenate its vector representations with the vectors from the first two baselines, in order to test how well models trained on real text and on graphs complement each other. The results are very encouraging, especially the combination with the **GoogleNews** model, which achieves by far the highest scores on the *WordSim* data sets and the closest results on *SimLex999* to those achieved via the hypernymy-inferred relations. While certainly there is a degree of redundancy in the vectors obtained via concatenation, the correlation improvements signal that there is also complementarity.

5 Neural Lexical Learning Using Graph Embeddings

This section presents two deep learning architectures for performing lexical analysis. The first one is an instance of now-standard recurrent neural network classifiers for sequence-to-sequence tasks, such as WSD and POS tagging (here called *Architecture A*), while the second does not optimize on a classification objective directly. Rather it tries to learn to represent context information in the same embedding space used to transform the input and gold data into numerical representations (here called *Architecture B*). The two architectures are described in greater detail below. They are both trained on the SemCor corpus and evaluated on the WSDEval data. Possible combinations of the two architectures and different tasks performed with them are discussed in connection with multi-task learning.

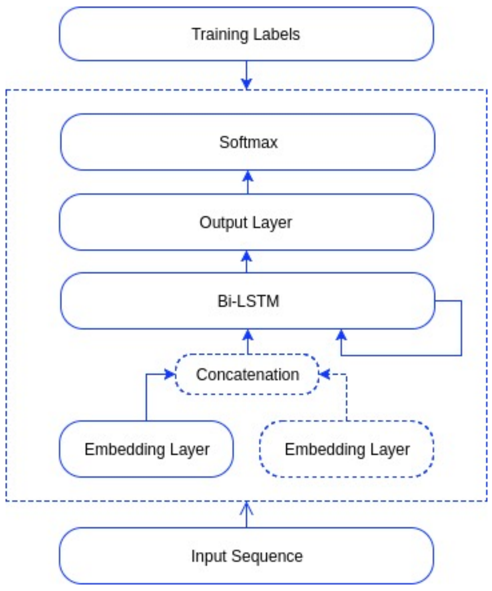

Figure 1: *Architecture A* – a recurrent neural network for sequence-to-sequence tagging. The dotted lines mean that a component or a connection is optional (in the case of concatenating embeddings from two different sources). Comparison between the training labels and the output is done via cross entropy.

Classification *Architecture A* is a Bi-LSTM network with an embedding layer at the input step and a softmax layer at the output (see fig. 1). When used for WSD, *Architecture A* maps the representation of a word context from its hidden layer(s) to a vector representing the synsets of all lemmas seen during training (WordNet 3.0 is used as dictionary). Words are disambiguated after consulting the dictionary to determine the possible synsets on the basis of lemma and POS information. Those not seen in training are disambiguated using WordNet's fist sense heuristic. Table 3 summarizes experimental work done with the architecture. Using the GloVe word embeddings (Pennington et. al., 2014), models trained with *Architecture A* achieve results within 1-2% accuracy from the state of the art, well above the most frequent sense baseline (typically difficult to beat in WSD). Experiments with additional VSMs whose embeddings are concatenated to the GloVe vectors do yield better results on separate data sets within WSDEval (Senseval-2, SemEval-13, SemEval-15), but they do not improve the purely GloVe-based result for the concatenation of all data sets, so they are not reported here (for more details and results see Popov (2017)). The model reported in the table has one hidden Bi-LSTM layer; the LSTM layers have sublayers of 200 neurons, each initialized from a random uniform distribution ([-1;1]); dropout of 20% is applied to the LSTMs; the learning rate is set to 0.2; training is done via stochastic gradient descent.

The same architecture can be parametrized for the POS tagging task – the only adaptation that needs to be done is changing the output and softmax layers, so that the network learns to classify according to the smaller set of possible tags. A multi-task learning setting that combines WSD and POS tagging requires merely two separate output layers, so that different probability distributions per tag set can be obtained. The cross-entropy losses from the two training signals are then summed and passed to the optimizer.

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
Architecture A	69.6	69.4	59.3	65.0	69.4	67.8
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	-	-	-
Architecture B	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 3: Comparison of the models trained with *Architectures 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 et. al. (2017a); IMS-2010 is reported in Zhong and Ng (2010); IMS-2016 (this is the configuration IMS + Word2Vec) 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.

Regression *Architecture B* is similar, but instead of using a softmax classifier, it compares the context representation (per word) to an embedding of the relevant gold label synset (see fig. 2). To that purpose, the hidden layer representation of the RNN is resized to the dimensionality of the VSM used to embed the inputs and to supply the synset embeddings. Least squares is used to compare the context vector and synset embeddings, the result of which serves for training. *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 having access to the embedding of 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, which provides an interpretation of the meaning of word senses. Combining *Architectures A* and *B* is done analogously to the previous setup, only in this case one of the computational pathways ends with a softmax layer, while the other – with a least squares comparison. *Architecture B* can also be used for WSD – via calculating the cosine distance between the context embeddings and potential synsets. It achieves results below the most frequent sense (MFS) baseline, but above most of the popular knowledge-based solutions (see table 3 and Raganato et. al. (2017a) for more KBWSD results).

As outlined above, 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.
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.

In order to obtain a VSM that combines words/lemmas and synsets in a shared space, the methodology from Goikoetxea et. al. (2015) is employed once again. The following parameters are used with the

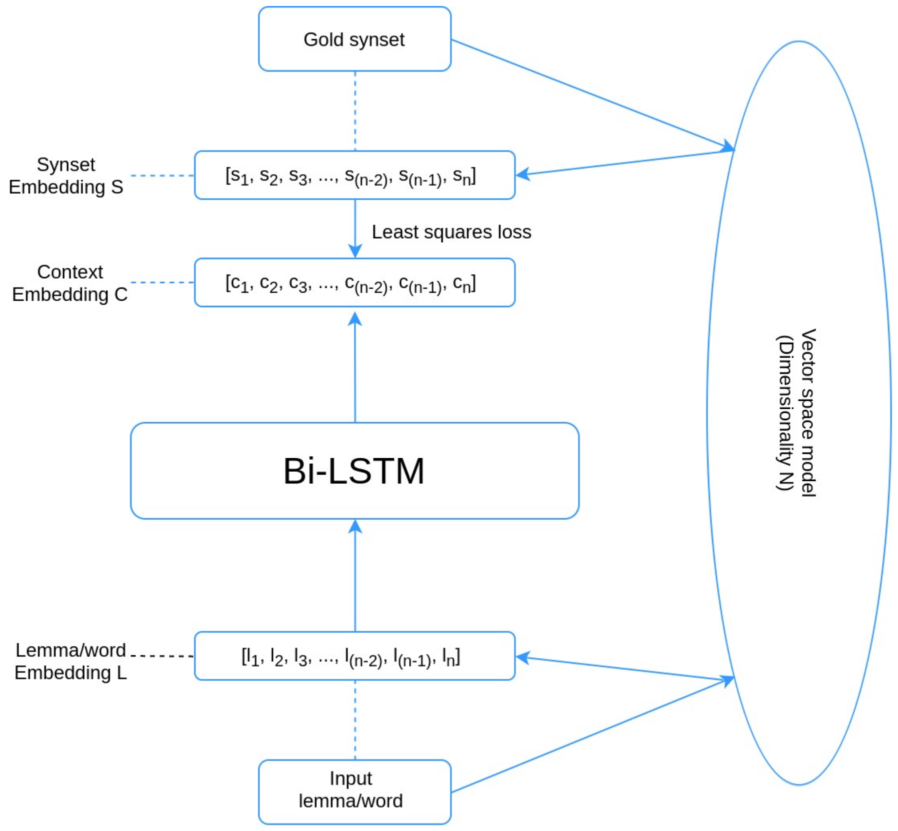
Gold synset
Synset Embedding S
$[s_1, s_2, s_3, \ldots, s_{(n-2)}, s_{(n-1)}, s_n]$
Least squares loss
Context Embedding C
$[c_1, c_2, c_3, \ldots, c_{(n-2)}, c_{(n-1)}, c_n]$
Bi-LSTM
Lemma/word Embedding L
$[l_1, l_2, l_3, \ldots, l_{(n-2)}, l_{(n-1)}, l_n]$
Input lemma/word
Vector space model (Dimensionality N)

Figure 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 backpropagated 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.

Word2Vec tool: number of training iterations = 7; number of negative samples = 5; window size = 15; threshold for the occurrence of words = 1e-7. The graph used for the generation of the pseudo-corpus – **WN30WN30glConOne** – is based on the original WordNet relations and is also further enriched with syntagmatic relations extracted from the gloss annotations in eXtended WordNet (see section 3 above). This experimental setup also concatenates a Wikipedia dump to the pseudo-corpus training data, so that functional words can be represented as well.

Next 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. The following parametrization of the network is used: 1 hidden Bi-LSTM layer; 400 units per sub-layer in the LSTMs; no dropout; optimization with stochastic gradient descent; learning rate of 0.2; random uniform initialization within [-1;1].

Table 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

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 4: Comparison of the models trained with *Architecture B* on the Senseval-2 data and using different VSMs. All models share the same parameters, 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 and Schütze (2015).

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, like SW2V and AutoExtend, do not seem to fare better than the 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-corpus-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-based 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.

6 Multi-task Lexical Learning

This section presents results on multi-task learning, where the experimental work is based on *Architectures A* and *B*. The first subsection discusses the combined training of a WSD classifier and a POS tagger, and the second one – of a WSD classifier and a context embedding model.

6.1 WSD and POS tagging within *Architecture A*

SemCor is used as training data, but this time the files with the original POS annotations are used[6], as opposed to the data from the Universal evaluation framework (UEF) by Raganato et. al. (2017a), used in the rest of the experiments. For evaluation, the Senseval-2 all-words data from the UEF set is used; the POS tagger run over it has been manually corrected for the open class words and therefore it is used here as a kind of silver resource. The final WSD still uses the gold POS tags at the post-processing step outside of the network, but the network is influenced by the training of the POS tagging branch. The GloVe vectors are used as input features. The multi-task solution achieves 0.5% higher accuracy on WSD and 1.2% higher accuracy on POS tagging, compared to the analogous single-task models (see table 5; note that the result on Senseval-2 is somewhat higher in this setup, but that is most probably due to some difference in the data set representations used for training).

6.2 WSD and context embedding. Combining *Architectures A and B*

SemCor is used for training and the all-words task data from Senseval and SemEval – for evaluation (all data is downloaded from the UEF). The mixed lemma&synset best-performing VSM from the previous section is used for the input and the gold synset representations. Table 6 summarizes the results. Both the classifier-based disambiguation and the context embedding modules make gains under multi-task learning. This is especially evident with respect to the context embedding system, which overcomes the

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

System	WSD (SNE-2)	POS (SNE-2)
Architecture A (single-WSD)	70.6	-
Architecture A (single-POS)	-	90.9
Architecture A (multi-task)	**71.1**	**92.1**

Table 5: 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”.

MFS baseline on some data sets. When the multi-task-trained classifier is regularized using dropout, it comes very near to the best result obtained with *Architecture A*. That best model uses the GloVe vectors, i.e. it has access to a VSM with representations of the input words obtained from a much larger data set. It is noteworthy that the GloVe vectors represent word forms, whereas the VSM used by the default models here encodes representations of lemmas only, i.e. it doesn't make any use of morphological information.

System	SNE-2	SNE-3	SME-07	SME-13	SME-15	ALL
Model A (single)	68.5	67.1	58.2	63.6	67.0	66.2
Model A (single, GloVe)	69.6	69.4	59.3	65.0	69.4	67.8
Model A (multi)	68.9	67.8	58.0	63.7	68.4	66.7
Model A (multi+dropout)	69.6	68.0	59.1	64.5	70.2	67.5
Model B (single)	64.7	57.9	47.9	61.9	64.8	61.3
Model B (multi)	66.8	60.1	49.2	63.4	67.7	63.3
MFS	65.6	66.0	54.5	63.8	67.1	64.8

Table 6: *Models A* perform *Architecure A*-style WSD, while *Models B* use the method described for *Architecture B*. The label ”multi” in parenthesis indicates that this is one of the two computational pathways of the multi-task model. All models are trained using the same parameters. The amount of dropout applied to the last of the *A-models* is 0.5. ”SNE” stands for ”Senseval”; ”SME” – for ”SemEval”.

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.

Analysis of the Results Here I offer a preliminary 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 have been excerpted from the *ALL* evaluation data set, together with the corresponding answers given by: the classification module (here called *A*), the context embedding 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 7 provides an overview of how often one or the other model is correct.

If it were the case that the type B modules (context embedding) are merely learning the same information as type A modules (classification), one would expect to find almost no examples where the context embedding module, and not the classification one, provides a correct answer. This would be especially true 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 context embedding architecture knows better than the classifier in many cases. In fact, it is more often correct in its predictions, by a wide margin. Note that this does not contradict the higher accuracy score of the classifier approach in general, as the latter uses a backoff heuristic (WN 1st sense) whenever it en-

Combination	A $\neq C \neq B$	B = C $\neq A$	A = C $\neq B$	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 7: 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 context embedding module picks the same answer as the WN 1st sense heuristic; and in the third one the classification module conforms to the WN 1st sense heuristic. ”A” stands for ”classification module”; ”B” – for ”context embedding 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.

counters a word it has not trained on. But if such cases are counted as errors on the part of the A models, then the context embedding module is clearly more powerful than the softmax-based part of the architecture. Model B is also leading the board in the cases 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 context embedding 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 types 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.

7 Conclusion and Further Work

This article has outlined a view of the lexicon as a model that combines different kinds of information pertaining to various NLP tasks and methods. It has demonstrated the usefulness of combining heterogeneous data and task objectives via a number of experimental setups. Experiments with knowledge-based WSD have shown that enriching a knowledge graph with syntagmatic information from corpora can result in significant advantage over baseline structured resources. Such enriched graphs can be beneficially exploited for the creation of vector space models that are able to beat popular corpus-based VSMs on standard evaluation tasks like word similarity and relatedness calculation. The VSMs can also be used as a source of features and training data for supervised deep learning architectures – for classification or regression tasks. And finally, multi-task learning is shown to provide significant gains when multiple objectives are combined, which depend on a common lexical representation.

The described models and algorithms will be integrated in the infrastructure developed within the CLaDA-BG project, and these promising results will be used as justification to investigate more complex models based on the interaction of different kinds of lexical interfaces. Some of the potential applications include: WSD (knowledge-based and supervised); text similarity (via context embedding); improved POS tagging.

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. More experimental work is necessary in order to determine which auxiliary tasks do and do not help with WSD (or other problems). This should be combined with integrative work to establish links and interfaces between existing lexical models (structured, symbolic, numerical). 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 constitute a serious step towards building multi-purpose and complexly structured linguistic and conceptual representations that resemble to a greater degree human cognition, rather than task-specific machinery.

Acknowledgements

This research has received partial support by the grant 02/12 – *Deep Models of Semantic Knowledge (DemoSem)*, funded by the Bulgarian National Science Fund in 20172019. In addition, some of the work has been done within the Bulgarian National Interdisciplinary Research e-Infrastructure for Resources and Technologies in favor of the Bulgarian Language and Cultural Heritage, part of the EU infrastructures CLARIN and DARIAH – *CLaDA-BG*, Grant DO01-164/28.08.2018. I am grateful to the anonymous reviewers for their comments and suggestions.

References

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: 19-27.

Agirre, E. and 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, Association for Computational Linguistics: 33-41.

Agirre, E., de Lacalle, O.L. and Soroa, A. 2018. The Risk of Sub-optimal Use of Open Source NLP Software: UKB is Inadvertently State-of-the-art in Knowledge-based WSD. arXiv preprint arXiv:1805.04277.

Alonso, H.M. and Plank, B. 2016. When is Multitask Learning Effective? Semantic Sequence Prediction Under Varying Data Conditions. arXiv preprint arXiv:1612.02251.

Bingel, J. and Søgaard, A. 2017. Identifying Beneficial Task Relations for Multi-task Learning in Deep Neural Networks. arXiv preprint arXiv:1702.08303.

Bonial, C., Stowe, K. and Palmer, M. 2013. Renewing and Revising SemLink. In Proceedings of the 2nd Workshop on Linked Data in Linguistics (LDL-2013): Representing and linking lexicons, Terminologies and Other Language Data: 9-17.

Camacho-Collados, J., Pilehvar, M.T. and 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: 567-577.

De Lacalle, M.L., Laparra, E. and Rigau, G. 2014. Predicate Matrix: Extending SemLink through WordNet Mappings. In LREC: 903-909.

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

Fellbaum, Christiane. 1998. Wordnet. Wiley Online Library.

Goikoetxea, J., Soroa, A. and Agirre, E. 2015. Random Walks and Neural Network Language Models on Knowledge Bases. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: 1434-1439.

Harabagiu, S. M. and Moldovan, D. I. 2000. Enriching the WordNet Taxonomy with Contextual Knowledge Acquired from Text. Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language, 301-333.

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. and Yu, K.. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. arXiv preprint arXiv:1508.01991.

Iacobacci, I., Pilehvar, M.T. and Navigli, R. 2015. Sensembed: Learning Sense Embeddings for Word and Relational Similarity. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers): 95-105.

Iacobacci, I., Pilehvar, M.T. and Navigli, R. 2016. Embeddings for Word Sense Disambiguation: An Evaluation Study. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers): 897-907.

Johansson, R. and 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: 1428-1433.

Kågebäck, M. and Salomonsson, H. 2016. Word Sense Disambiguation Using a Bidirectional lstm. arXiv preprint arXiv:1606.03568.

Kiros, R., Zhu, Y., Salakhutdinov, R.R., Zemel, R., Urtasun, R., Torralba, A. and Fidler, S. 2015. Skip-thought Vectors. In Advances in Neural Information Processing Systems: 3294-3302.

Levin, Beth. 1993. English Verb Classes and Alternations: A Preliminary Investigation. University of Chicago Press.

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

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

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

Mihalcea, R. and Moldovan, D. I. 2001. eXtended WordNet: Progress Report. In Proceedings of NAACL Workshop on WordNet and Other Lexical Resources: 95100.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In Advances in Neural Information Processing Systems: 3111-3119.

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

Miller, G.A., Leacock, C., Tengi, R. and Bunker, R.T. 1993. A Semantic Concordance. In Proceedings of the Workshop on Human Language Technology, Association for Computational Linguistics: 303-308.

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

Nguyen, D.Q., Dras, M. and Johnson, M. 2017. A Novel Neural Network Model for Joint POS Tagging and Graph-based Dependency Parsing. arXiv preprint arXiv:1705.05952.

Palmer, M. 2009. Semlink: Linking Propbank, Verbnet and FrameNet. In Proceedings of the Generative Lexicon Conference, Pisa, Italy: GenLex-09: 9-15.

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

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

Popov, A. 2017. Word Sense Disambiguation with Recurrent Neural Networks. In Proceedings of the Student Research Workshop associated with RANLP: 25-34.

Raganato, A., Camacho-Collados, J. and Navigli, R. 2017. Word Sense Disambiguation: A Unified Evaluation Framework and Empirical Comparison. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers: 99-110.

Raganato, A., Bovi, C.D. and Navigli, R. 2017. Neural Sequence Learning Models for Word Sense Disambiguation. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: 1156-1167.

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

Ruder, S., 2017. An Overview of Multi-task Learning in Deep neural Networks. arXiv preprint arXiv:1706.05098.

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

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

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: 596-603.

Wang, P., Qian, Y., Soong, F.K., He, L. and Zhao, H. 2015. 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. and Zhao, H. 2015. A Unified Tagging Solution: Bidirectional LSTM Recurrent Neural Network with Word Embedding. arXiv preprint arXiv:1511.00215.

Zhong, Z. and 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, Association for Computational Linguistics: 78-83.