The Impact of Word Embeddings on Neural Dependency Parsing

Benedikt Adelmann
Wolfgang Menzel
Fachbereich Informatik
Universität Hamburg
adelmann@informatik.uni-hamburg.de
menzel@informatik.uni-hamburg.de

Heike Zinsmeister
Institut für Germanistik
Universität Hamburg
heike.zinsmeister@uni-hamburg.de

Abstract

Using neural models to parse natural language into dependency structures has improved the state of the art considerably. These models heavily rely on word embeddings as input representations, which raises the question whether the observed improvement is contributed by the learning abilities of the network itself or by the lexical information captured by means of the word embeddings they use. To answer this question, we conducted a series of experiments on German data from three different genres using artificial embeddings intentionally made uninformative in different ways. We found that without the context information provided by the embeddings, parser performance drops to that of conventional parsers, but not below. Experiments with domain-specific embeddings, however, did not yield additional improvements in comparison to large-scale general-purpose embeddings.

1 Introduction

In recent years, using neural models has notably improved the accuracy of dependency parsing, compared to non-neural or 'conventional' statistical parsers. However, while typical non-neural parsers normally have to extract all knowledge encoded in their models, including lexical information, from the training data, i. e. a dependency treebank, neural dependency parsers are usually endowed with word embeddings in addition to the treebank, not only at training, but also at test time. Given that embeddings are highly informative about distributional properties of the embedded entities (words in this case), which probably correlate with the possibility or plausibility of syntactic relationships, and that they are generally trained on corpora orders of magnitude larger than the dependency treebanks available for any language, this can be seen as an additional external source of information that conventional parsers do not have at their disposal.

This raises the question of how much of the reported difference, if any, is due to the neural model being better at modelling syntax and how much is just due to the information in the embeddings. On the one hand, one could argue that this distinction is irrelevant because the comparison reflects the way the systems would be used in practice. On the other hand, however, it is scientifically unsound to derive claims about capability differences of models or formalisms from experiments where more than just the model or formalism changes with respect to a control setting.

Furthermore, insight into the individual influence of model parts on the overall output (or at least its quality) can be seen as a step towards (some kind of) interpretability. Understanding the influence of embeddings is especially useful in language processing, where most knowledge is symbolic while neural networks necessarily operate on continuous representations. As it is embeddings of some kind that bridge this gap, systems should not be too dependent on their quality.

To gain more insight into this dependence of dependency parsing on embeddings, we have conducted experiments with a neural dependency parser provided with deterministically uninformative as well as random word embeddings and we report on the results.

2 Related Work

To our knowledge, the mechanisms leading to neural parsers exhibiting better performance than conventional ones have not yet been investigated. It has been shown that recurrent neural networks are able to capture syntactic structures such as nesting in practice as long as the depth is bounded (Bhattamishra et al., 2020), but this does not make a statement about whether or why they are better at it than conventional parsers, and it remains unclear what influence the input embeddings have on this capability.

The question how a model output changes when trained and evaluated on different input embeddings, specifically word embeddings, has been addressed by Rios and Lwowski (2020). They train

numerous word embeddings using Word2Vec, GloVe or fastText, each with various different initialization seeds and on different corpora, and compare the performance of models when using these different embeddings as input. We take a similar approach, except that we use 'artificial' embeddings, and while their focus is on the consequences of embedding differences due to algorithm and initialization, we are interested in the impact of the (distributional) semantics available through the embedding in the first place.

For a short period in time there were even some neural parsing architectures without (external) embeddings, such as the ISBN parser by Titov and Henderson (2007). Its reported performance was well below what current parsers (with external embeddings) achieve, similar indeed to that of non-neural parsers.

Within a more recent apporach, parsing performance with and without external word embeddings has been compared by Kiperwasser and Goldberg (2016), who mention a counter-intuitive finding that external word embeddings *degraded* the performance of one of their parsers. In the small ablation study they report, however, the addition of external embeddings was accompanied by a change in parsing strategy (from graph-based to greedy transition-based), not allowing for conclusions about the impact of the embeddings alone.

More generally, there has been growing interest in the relationship between embeddings and downstream tasks in recent years, usually with a focus on the knowledge possibly encoded in the embedding, but also on how this knowledge and its representation affect further processing to which it is used as input. Much work on this topic has been concerned with sentence embeddings; for example, Miaschi et al. (2020) find a correlation between the amount of linguistic knowledge represented in a sentence embedding and its ability to solve a specific downstream task. They also provide evidence that fine-tuning the embedding makes it represent more task-specific knowledge at the expense of general knowledge.

A popular method for assessing what linguistic knowledge an embedding represents is *probing tasks* (a term that seems to have been coined by Conneau et al., 2018, based on Adi et al., 2017, and Shi et al., 2016), classifiers trained to reconstruct known explicit linguistic properties from embeddings. In one sense, dependency parsing can be seen as a probing task where the linguistic property to be extracted is the dependency structure of a sentence, and has indeed been used as a probing task (Miaschi et al., 2020; Kunz and Kuhlmann, 2020). However, 'viewing probing results in isolation can lead to overestimating the linguistic capabilities of a model' (Mosbach et al., 2020, p. 780), and Kunz and Kuhlmann (2020) point out that in such scenarios, it is generally unknown to what extent the output is indeed present in and extracted from the embedding, as opposed to being learned by the model ('probe') built on top of it. They consider embeddings to most likely lie between two extremes: no useful information being represented at all, or the information already being represented in a human-readable way. Apart from restricting the probing classifier to limited expressiveness, one possibility of distinguishing embedding from classifier power is therefore the comparison with the results of probing baseline embeddings lacking *any* linguistic information content, a common choice being random ones. We too use randomness as one way to create such embeddings.

A study relating word-level probing tasks to higher-level processing for several languages, including dependency parsing for German, can be found in Şahin et al. (2020). They report significant correlations between dependency parsing and morphosyntactic probing performance, suggesting that not only semantic, but also morphosyntactic information encoded in a word embedding can be influential. Note though that neural dependency parsing based on word embeddings is different from probing sentence embeddings for dependencies of the encoded sentence. One could say that the situation is the converse: In the probing scenario, the embedding is the result of a procedure and is probed to investigate its dependence on the original input. In our case, the embeddings are the input, and we want to investigate the dependence of the procedure on it. There are similar findings to the above for word embeddings, due to Köhn (2016), attesting the choice of embeddings a noticeable impact on parser performance.

3 Experimental Setup

As we cannot directly inspect what the neural architecture learns and whether it is indeed better than 'conventional' (non-neural) architectures at learning the syntactic knowledge needed for parsing, we employ a proxy question instead and ask

how the output of a neural parser changes when depriving it of the knowledge encoded in the input word embeddings, as these embeddings are an additional input that most conventional parsers do not have at their disposal. If the neural parser performs significantly better than conventional parsers when provided with the same input, its neural architecture is obviously a better learner of syntax than the architectures of the conventional parsers. On the other hand, if the neural parser needs more input (i. e. the embeddings) than the conventional parsers to outperform them, the comparison is inherently unfair as it is hardly surprising that a system with more input can yield better predictions. While this does not necessarily rule out the possibility that the neural architecture is superior, the performance impact of eliminating a source of information sheds light on the dependence on that information. Such a dependence may be undesirable in certain contexts, such as low-resource settings where high-quality word embeddings are unavailable.

Another common scenario is that of *domain adaptation*, where only a generic treebank of considerable size is available for training, but specific embeddings can be obtained in an unsupervised[1] way from in-domain data (possibly the same data one wishes to parse later), which may be much smaller than the data employed for training general-purpose embeddings. We complement our experiments on the impact of uninformative embeddings by also providing the parser with embeddings trained on the corpora from which we draw our test data.

3.1 Parser

The parser we experiment with is **Sticker** (de Kok and Pütz, 2020), a recent neural dependency parser treating parsing as a sequence labelling problem: Every token is assigned a complex tag encoding where to attach it. In the case of Sticker, the tags indicate the attachment point as its relative position among tokens with a part of speech (e. g. 'the second finite verb to the left') and are computed by a neural network. (From the different architectural options we chose the LSTM architecture, which had turned out to work best on our data.) The only information that the neural network is provided with as input are embedding vectors of the tokens (words) in the sentence and of their part-of-speech (POS) tags. At training time, the parser trains the network based on these inputs (and the gold dependency structure and labels), but it does not alter the embeddings provided nor save any other lexical information about words in the training data; in particular, there is no attempt to obtain semantic knowledge about words not covered by the embedding.[2] This implies a substantial dependence on those embeddings.

As a conventional baseline we employ the five non-neural parsers from Adelmann et al. (2018a), excluding JWCDG, but only report the performance of the best parser per test text as reference. In all cases this was either Malt[3] (Nivre, 2003) with the 'Covington non-projective' algorithm (Covington, 2001) or Mate[4] (Bohnet, 2010).

3.2 Uninformative Embeddings

As Sticker cannot be run without word embeddings as input, we cannot entirely turn off this input, but we can substitute artificially created pseudo (or 'dummy') embeddings that are 'uninformative' in the sense that they do not encode any properties of the words beyond the word form identity (in particular, no semantics at all). We experiment with such uninformative embeddings created in different ways, two of them deterministic and four random (sampled with respect to different distributions, thus having different properties):

empty: an embedding not containing any words at all. This will make any word form encountered by the parser out-of-vocabulary (just like rare word forms simply not covered by a 'normal' embedding).

zero: an embedding mapping every word to the zero vector (the vector containing only zeroes). The out-of-vocabulary words are therefore the same as for the informative control embedding (see further below), but as all of them are assigned the same vector, they are entirely indistinguishable when processing them only by means of their word vectors.

cube: an embedding mapping every word to a vector with stochastically independent components all uniformly distributed in the unit interval $[0, 1)$. In contrast to the previous embedding, words now have different vectors and are therefore

[1] Or 'self-supervised', referring to the fact that manual annotation effort is unnecessary.

[2] Details given here that are not from the cited paper are from personal communication with Daniël de Kok.

[3] http://www.maltparser.org/

[4] https://code.google.com/archive/p/mate-tools/

distinguishable, but as the vectors are chosen at random, they are highly unlikely to correlate with any linguistic relation: They do not carry any semantic information whatsoever.

ccube: like *cube*, but shifted into the origin, i. e. with components drawn from $[-0.5, 0.5)$.

gauss: an embedding mapping every word to a standard normal random vector, i. e. a vector with stochastically independent components all following a standard normal ('Gaussian') distribution.

sphere: an embedding mapping every word to a vector of length one (i. e. on the Euclidean unit sphere, hence the name), with every such vector having equal probability. In this embedding, any word vector can be separated from every other word vector by some hyperplane, so distinguishing words should be especially easy.

As an informative control embedding we use the German word embedding released with the pre-trained Sticker models.[5] Except for 'empty', which does not contain any vectors at all, all artificially created embeddings share dimension (300) and vocabulary with the control embedding.

For testing the influence of domain-specific embeddings, we train additional embeddings on texts sampled from the test corpora (see Section 3.4).

As mentioned above, the parser also requires an embedding of the part-of-speech (POS) tags present in the input. The control embedding here is based on one released with the pre-trained Sticker models which embeds the STTS (Schiller et al., 1999).[6] Additionally we created uninformative embeddings of the same six types as above, again with vocabulary (tag inventory; except for 'empty') and dimension (50) the same as in the control embedding.

However, we do not provide the parser with *both* uninformative word *and* uninformative POS embedding, as the only input that the parser receives are embedded words and POS tags, so making both embeddings uninformative would actually decouple the parser from its input.[7] We have not tried combining the uninformative POS embeddings with the domain-specific word embeddings either.

This leaves us with four types of neural parser configuration: With control word and control POS embedding (baseline), with uninformative word and control POS embedding, with control word and uninformative POS embedding, and with domain-specific word and control POS embedding.

For every artificial embedding we train one model for the parser and the respective embedding on the first 91,999 sentences of part A of the *Hamburg Dependency Treebank* (Foth et al., 2014), with the remaining 10,000 sentences (9.8 %) as validation set.

3.3 Test Data

To obtain test data, three annotators manually annotated randomly drawn sentences from three different corpora. The first one is a corpus of 636 modern dystopias written by German writers. The second one is the *d-Prose* corpus (Gius et al., 2020) containing 2,529 literary German prose texts from between 1870 and 1920. The third one consists of 8,788 documents downloaded from the internet, selected by the appearance of German keywords related to telemedicine (Franken and Adelmann, 2021). The sentences sampled from each corpus were combined with the annotated sentences of the respective texts from Adelmann et al. (2018b). The three test sets comprise around 7,500 tokens and 450 sentences each, with similar sentence length distributions (for details see Table 5; this, as well as some other tables, can be found in the appendix).

These datasets can be expected to notably differ both stylistically and thematically from the training data and between each other, without being intrinsically hard to annotate (and parse) like spoken or Twitter data.

The three annotators annotated the texts with dependency relations following the guidelines of Foth (2006), obtaining an overall inter-annotator reliability of Fleiss' $\kappa = 0.89$ for unlabelled attachment accuracy and Fleiss' $\kappa = 0.93$ for labelled attachment accuracy on a balanced subset of about 20 % of the test data. The remaining data

[5] German word embeddings, trained on TüBa-D/DP (de Kok and Pütz, 2019), quantized using optimized product quantization: https://github.com/stickeritis/sticker-models/releases/tag/de-structgram-20190426-opq (September 16, 2019, last retrieved April 14, 2021)

[6] With PAV instead of PROAV; source of the original embedding: https://blob.danieldk.eu/sticker-models/de-structgram-tags-20190426.fifu (last retrieved May 14, 2021)

[7] As a sanity check we did try that, obtaining UAS values between 17 % and 22 % and LAS values between 10 % and 16 %. Note that even in this scenario the parser still has access to sentence lengths, and POS tags are available when determining the attachment point based on the complex tag being predicted by the neural network, which itself does not have this information.

was distributed among the annotators (so that sentences were annotated by only one annotator each) and subsequently post-edited based on some heuristics for checking consistency.

The annotators only annotated dependencies. POS tags (required by all parsers as input), lemmata and morphological features (required by the non-neural parsers) were predicted by a tagger ensemble.[8] This is in contrast to training time, where gold POS tags from the treebank were used.[9]

3.4 Domain-Specific Embeddings

To obtain domain-specific embeddings we trained word embeddings on samples of similar total token count as part A of the *Hamburg Dependency Treebank* (approx. 1,872,622 tokens) from each of our test corpora, a reasonable order of magnitude for domain-specific data. The samples were chosen at random from the test corpora, taking care that no sentences used as test data were also selected as training data for the embeddings. Additionally, we sampled a collection of sentences, again of roughly the same total token count, from the union of all three test corpora.

4 Results

We assess performance differences by comparing unlabelled and labelled attachment accuracy (also known as unlabelled and labelled attachment score, or UAS and LAS) with respect to our test data between the best conventional (non-neural) parser, the neural parser with the ('informative') control embeddings, and the neural parser with our manipulated (i. e. uninformative or domain-specific) embeddings. For the webcrawling data, the best-performing conventional parser was Malt; for the other test sets, it was Mate.

Usually, such attachment accuracies are computed *excluding punctuation* since punctuation attachment and labelling is considered trivial. This, however, may not be the case if uninformative embeddings make it hard for the parser to determine which tokens are in fact punctuation. For this reason, we treat punctuation like any other tokens and report attachment accuracies *including punctuation*. Between 12 % and 17 % of the tokens in our test data are punctuation (according to automatic POS tagging), so they also increase the effective amount of test data, and when excluding them, attachment accuracies are about 2 percentage points lower than those we report, for both the neural and the conventional baseline.

4.1 Uninformative Word Embeddings

With the control embedding, the neural parser has a UAS 3 to 4 percentage points higher than the best conventional parser and an LAS 5 to 6 percentage points higher; this is a considerable baseline difference. With uninformative word embeddings, this margin decreases by 1 to 3 percentage points in the case of UAS and by 1 to 7 percentage points for LAS, depending on test set and the type of uninformative embedding. For instance, on the modern dystopias data with the 'cube' embedding, the UAS decreases from 0.93 to 0.90, and the LAS decreases from 0.91 to 0.84, the UAS reducing to and the LAS even falling short of Mate's performance (cf. Table 1). The other uninformative embeddings have less dramatic effects, giving values generally still above the conventional baseline. For all test sets, the embedding with the highest UAS and LAS is 'sphere', and the 'cube' embedding is among those with the smallest UAS and LAS. The other embeddings do not differ much from each other, their accuracies being mostly closer to those of the conventional than those of the neural baseline. Performance differences between test sets are similar for the baseline models (both conventional and neural) and the models with uninformative embeddings.

As the UAS and LAS differences are small, we also tested for statistical significance, using the randomization test of Yeh (2000) (with 100,000 samples) because theoretical distributions are not known. Except for the 'sphere' embedding tested on webcrawling data or the combination of all three, the *p*-value for the hypothesis that the model performs as well as the *neural* baseline is below 5 %; in the vast majority of cases, it is even below the stricter significance threshold of 0.25 % proposed by Søgaard et al. (2014), so we can be confident that the models do indeed perform *worse* than the neural baseline. On the other hand, the *p*-value for the hypothesis that the model performs as well as the *conventional* baseline is mostly not below the strict threshold, but below 5 % in more than half of the cases (see Table 4). The hypothesis cannot be rejected for the UAS of the 'cube' embedding (i. e. this embedding makes the neural parser perform no better than the best conventional parser, at least not with respect to

[8] See https://github.com/benadelm/hermA-Pipeline (last retrieved August 7, 2021).

[9] Again, with PAV instead of PROAV.

text	traditional			normal		empty		zero		cube		ccube		gauss		sphere	
	Parser	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
dystopias	Mate	0.90	0.85	0.93	0.91	0.91	0.87	0.91	0.87	0.90	0.84	0.91	0.86	0.90	0.86	0.92	0.88
19th century	Mate	0.88	0.83	0.91	0.88	0.89	0.84	0.89	0.84	0.88	0.83	0.89	0.84	0.88	0.84	0.90	0.85
webcrawling	Malt	0.87	0.83	0.91	0.88	0.88	0.85	0.88	0.85	0.88	0.84	0.89	0.86	0.88	0.86	0.90	0.87
all three	Mate	0.88	0.83	0.92	0.89	0.90	0.85	0.89	0.85	0.89	0.84	0.90	0.85	0.89	0.85	0.91	0.87

Table 1: Attachment accuracies for the uninformative word embeddings, including punctuation

text	traditional			normal		empty		zero		cube		ccube		gauss		sphere	
	Parser	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
dystopias	Mate	0.90	0.85	0.93	0.91	0.91	0.87	0.91	0.87	0.92	0.89	0.93	0.90	0.93	0.90	0.93	0.90
19th century	Mate	0.88	0.83	0.91	0.88	0.89	0.85	0.90	0.86	0.91	0.86	0.91	0.88	0.91	0.88	0.91	0.88
webcrawling	Malt	0.87	0.83	0.91	0.88	0.89	0.87	0.89	0.87	0.91	0.88	0.91	0.88	0.91	0.88	0.91	0.88
all three	Mate	0.88	0.83	0.92	0.89	0.90	0.87	0.90	0.87	0.91	0.88	0.92	0.89	0.92	0.89	0.92	0.89

Table 2: Attachment accuracies for the uninformative POS embeddings, including punctuation

text	traditional			normal		dystopias		19th century		webcrawling		total	
	Parser	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
dystopias	Mate	0.90	0.85	0.93	0.91	0.93	0.90	0.93	0.90	0.93	0.90	0.93	0.90
19th century	Mate	0.88	0.83	0.91	0.88	0.90	0.87	0.91	0.88	0.91	0.87	0.91	0.87
webcrawling	Malt	0.87	0.83	0.91	0.88	0.90	0.88	0.91	0.88	0.91	0.88	0.90	0.87
all three	Mate	0.88	0.83	0.92	0.89	0.91	0.88	0.91	0.89	0.91	0.88	0.91	0.88

Table 3: Attachment accuracies for the domain-specific word embeddings, including punctuation

head attachments), but it can be rejected (even with the stricter threshold) for the 'sphere' embedding (i. e. this embedding makes the neural parser still perform better than the best conventional parser). For the other embeddings, the picture is mixed. Even where the *p*-value is below 5 %, it is not much lower, so one should be cautious about rejecting the null hypothesis.

4.2 Uninformative POS Embeddings

For the uninformative POS embeddings, UAS and LAS values are higher than for the uninformative word embeddings. The 'ccube', 'gauss', and 'sphere' embedding even result in the same UAS as the control embedding (and so does the 'cube' embedding on the 19th century and web-crawling data). This is not very surprising since there are substantially fewer POS tags than words, and consequently, uninformative POS embeddings mean less information loss than uninformative word embeddings. Still, performance decreases with respect to the baseline can be observed over all test sets for the 'empty' and 'zero' embeddings, and for the other uninformative embeddings, there seems to be a tendency towards reductions in LAS (see Table 2). The increase in UAS from uninformative word to uninformative POS embeddings is smaller (1.6 percentage points on average) than the increase in LAS (2.6 percentage points on average), suggesting that there are in comparison more label errors when word embeddings are uninformative than when only POS embeddings are. Additionally, all values across the board are better now than those of the conventional parsers.

Correspondingly, *p*-values (Table 9) do clearly not permit rejection of the hypothesis that the uninformative 'ccube', 'gauss', or 'sphere' embedding makes the neural parser perform worse than with the control embedding, and the hypothesis that the performance is only as good as that of the conventional baseline can be rejected to the strict significance level of 0.25 %. The latter is even true for the 'cube' embedding, while the *p*-value for the test against the neural baseline LAS is also below 0.25 % for the dystopias and still below 5 % for the 19th century novels. The 'empty' and 'zero' embeddings exhibit mixed values. The *p*-values are below 5 % when testing against either baseline (but mostly not below 0.25 % for the neural baseline), with values below the stricter threshold appearing mostly for the LAS against the conventional parsers. Hence, here the assertion that the neural parser yields a better LAS than the conventional ones even with uninformative POS embeddings is more likely true than the corresponding one about the UAS. Apparently uninformative word embeddings have a stronger negative impact on LAS than uninformative POS embeddings.

4.3 Domain-Specific Word Embeddings

A difference between the neural parser's performance with domain-specific word embeddings and

text	empty		zero		cube		ccube		gauss		sphere	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
dystopias	***0.0430***	***0.0010***	***0.0170***	***0.0010***	***0.0010***	***0.0010***	***0.1360***	***0.0010***	***0.0010***	***0.0010***	*3.0180*	***0.0190***
19th century	***0.0960***	***0.0010***	***0.1520***	***0.0010***	***0.0110***	***0.0010***	*0.7470*	***0.0010***	***0.0130***	***0.0010***	*3.1170*	***0.0910***
webcrawling	***0.0200***	***0.0080***	***0.0040***	***0.0010***	***0.0040***	***0.0010***	*0.4580*	*0.3800*	***0.0160***	***0.0840***	7.0029	*3.3850*
all three	*0.7350*	***0.0210***	*0.5000*	***0.0150***	***0.0570***	***0.0010***	*2.5480*	***0.0410***	***0.2090***	***0.0100***	10.1369	*1.8220*

(a) *p*-values for the hypothesis that the results are not worse than Sticker's performance

text	empty		zero		cube		ccube		gauss		sphere	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
dystopias	*4.5430*	*1.4080*	8.9129	*2.0470*	26.1857	*2.8720*	*2.4620*	10.8949	44.2846	35.8266	***0.1020***	***0.0020***
19th century	*4.8630*	10.7799	*3.7090*	9.4529	17.6328	38.7696	*1.3280*	12.4229	14.2029	21.2568	*0.3060*	***0.0840***
webcrawling	*1.5910*	*0.3840*	*3.4910*	*1.8410*	6.4879	*3.0440*	***0.1160***	***0.0120***	*1.8230*	***0.0450***	***0.0040***	***0.0020***
all three	5.8059	*3.7340*	7.2999	*4.9590*	25.4797	41.4556	*2.2270*	*3.7790*	15.4428	8.3229	*0.3610*	***0.0740***

(b) *p*-values for the hypothesis that the results are not better than the performance of the respective best conventional parser (see Table 1)

Table 4: *p*-values (in %) for Yeh's randomized permutation test on performance differences between the uninformative word embeddings and the two baselines. Values below the significance threshold of 5 % are marked in italics; values below the stricter threshold of 0.25 % are additionally marked in bold. Values for the combination of all three corpora were computed on a subset of 461 sentences so that *p*-values are comparable.

with the control embedding is almost nonexistent, and the *p*-values are never below the significance threshold either. Conversely, they are always below the strict threshold for the hypothesis that the performance is not better than that of the conventional parser. While it is notable that even 'little' data the size of a dependency treebank (embeddings are usually trained on much bigger corpora) are sufficient to create an embedding sufficiently informative for the parser,[10] this does so far not facilitate insight into the role the embedding may play in domain adaptation. We did not test for effects of the embeddings being used cross-domain (e. g. the embedding trained on 19th century novels being used for parsing web-crawling data) as the performance differences among the different embeddings for the same test set are small where present at all, so we expect differences between test sets to be largely due to other parser-challenging aspects (such as general sentence complexity).

4.4 Label-Specific Evaluation

Finally, we take a brief look at some individual dependency labels. As pointed out in Adelmann et al. (2018a), overall attachment accuracies are skewed towards the performance on frequent phenomena such as determiner attachment, obfuscating issues with dependency relations that are of interest to content analyses, but appear less often. This evaluation only refers to the combination of all three test sets in the hope that as many labels as possible will be frequent enough there to be meaningfully evaluated.

For a number of labels, attachment precision and recall changed by more than 10 percentage points when parsed with uninformative embeddings, compared to parsing with the control embedding. Out of those, eleven appear more than 100 times in our test data; Table 11 shows their attachment precision and recall. Similarly great or in some cases even greater differences can also be observed for eleven other labels, but those are less frequent, some of them indeed very infrequent (e. g. there are only four occurrences of OBJG), so their values are probably unreliable. Among the frequent labels, heavy losses (up to 56 percentage points) can be observed for OBJD (dative object) and OBJP (prepositional object), mainly for the 'empty', 'zero' and 'cube' embeddings. OBJA (accusative object), PRED (predicative) and GMOD (genitive modifier) show losses mainly for these three embeddings, too, albeit not as big. With the 'ccube', 'gauss' and 'sphere' embeddings, losses are generally smaller, and for KOM (comparison), recall even *rises* with the 'ccube', 'gauss' and 'sphere' embedding.

There are also three labels where almost no difference in precision and recall can be observed for the deterministically uninformative embeddings ('empty' and 'zero'), but for the other (the random) embeddings: APP (apposition), ROOT and S. The latter two are especially interesting as S denotes the root node of sentences (in HDT, this is usually the finite verb) and ROOT is the label used exclusively for punctuation. While the precision of ROOT is

[10] We have not tested how well the domain-specific embeddings capture relationships between the embedded words.

always 1.00 (when the parser assigns this label, it is always correct), recall drops from almost 1.00 by 13 to 14 percentage points for the 'cube', 'ccube' and 'gauss' embeddings, that is, with those embeddings the parser fails to correctly identify about 13 to 14 % of the punctuation tokens. This is strange and remarkable given that punctuation is trivially identified by its POS. The decrease does not occur for the deterministic embeddings, nor for 'sphere'. S exhibits a similar phenomenon, but there it is precision that drops while recall remains, meaning that the parser mis-identifies something as a sentence root.

Table 12 shows precision and recall when parsing with uninformative POS embeddings, for the same labels as above. As with UAS and LAS, differences are less pronounced here, except for three labels when parsing with the deterministic embeddings: KOM shows a considerable increase in recall and OBJI (object infinitive) in precision, while ROOT decreases, again by 13 percentage points. This is complementary to the situation with uninformative word embeddings, where ROOT does not decrease for these two embeddings.

For the sake of completeness we note that there were no particularly interesting label performance differences when parsing with the domain-specific embeddings (Table 13).

5 Conclusion and Future Work

The main motivation for this paper was the question of whether neural networks are better than conventional, non-neural architectures at learning the syntactic knowledge needed for parsing, as opposed to just having the advantage of being provided with extra information in the form of word embeddings, and we approached this using the proxy question of how the output of a neural parser changes when depriving it of this extra information. The answer to this question from our results can be framed in two ways, depending on the perspective: Even without access to the knowledge encoded in a word embedding, the neural parser still performs (at least) as well as the best non-neural parser, so this lack of knowledge does not impair it so much that a conventional tool would be clearly preferable. Or alternatively: Without access to the knowledge encoded in a word embedding, the neural parser performs only about as well as the best non-neural parser, implying that it may indeed very well be the knowledge in the embedding that enables superior performance, not a superiority of the architecture.[11]

The results further suggest that a lack of word embedding knowledge abets label errors, while a lack of POS embeddings abets attachment errors, with a general tendency towards an increase in label errors in both cases. This could mean that knowledge about the co-occurrence of POS tags is more useful for predicting the correct head and knowledge about the co-occurrence of words is more useful for choosing the correct dependency label, which would not be implausible from a linguistic point of view. More dedicated experiments are necessary, however, to corroborate this hypothesis.

We also found that not all dependency labels are affected equally, the losses being concentrated mainly at 'content-related' labels such as OBJA (accusative object), with the especially vexing observation that uninformative word embeddings hinder the correct labelling of punctuation even though POS information should be sufficient to do so. A qualitative analysis of the label errors could be illuminative; possible reasons for this oddity would have to be investigated in greater depth.

The experiment with domain-specific embeddings was inconclusive, at least with the limited amount of domain-specific data used; the differences in vocabulary and in word semantics between the corpora were possibly too small to have a noticeable impact on parsing. We do observe, though, that even embeddings trained on little data make the parser perform almost as well as the control embeddings trained on big data.

Given this finding, subsequent research would have to dig further into the relationship between the size of the data used for training word embeddings and parser performance when using them.

We conducted our experiments with only one single parser. To assess how well our results apply to neural dependency parsing in general, future work would have to examine other parsers as well, particularly ones built on other parsing paradigms such as transition-based or graph-based parsing. It could furthermore be insightful to draw a comparison with conventional parsers able to use word embeddings (e. g. RBGParser).

[11] Of course, the mere ability to utilize word embeddings can be seen as an architectural superiority. This is not restricted to neural networks, though: RBGParser (Lei et al., 2014), too, can use word embeddings (cf. Köhn, 2016).

Acknowledgements

This work has been funded by 'Landesforschungsförderung Hamburg' in the context of the *hermA* project (LFF-FV 35). We would like to thank the reviewers for their thorough comments, Lea Röseler and Emily Roose for their invaluable annotation effort and Piklu Gupta for improving our English. All remaining errors are ours.

References

Benedikt Adelmann, Melanie Andresen, Wolfgang Menzel, and Heike Zinsmeister. 2018a. Evaluation of Out-of-Domain Dependency Parsing for its Application in a Digital Humanities Project. In *Proceedings of the 14th Conference on Natural Language Processing (KONVENS 2018)*, pages 121–135.

Benedikt Adelmann, Melanie Andresen, Wolfgang Menzel, and Heike Zinsmeister. 2018b. Manual Dependency Annotation of Three German Text Extracts from the Project hermA (Gold Standard Data). doi:10.5281/zenodo.1324079.

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained Analysis of Sentence Embeddings Using Auxiliary Prediction Tasks. In *Proceedings of ICLR Conference Track*, Toulon, France.

Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. 2020. On the Practical Ability of Recurrent Neural Networks to Recognize Hierarchical Languages. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1481–1494.

Bernd Bohnet. 2010. Very High Accuracy and Fast Dependency Parsing is not a Contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 89–97.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136.

Michael Covington. 2001. A Fundamental Algorithm for Dependency Parsing. In *Proceedings of the 39th Annual ACM Southeast Conference*, pages 95–102.

Kilian Foth. 2006. Eine umfassende Constraint-Dependenz-Grammatik des Deutschen. Technical report, Universität Hamburg.

Kilian Foth, Arne Köhn, Niels Beuck, and Wolfgang Menzel. 2014. Because Size Does Matter: The Hamburg Dependency Treebank. In *Proceedings of the Language Resources and Evaluation Conference 2014*. European Language Resources Association (ELRA).

Lina Franken and Benedikt Adelmann. 2021. Webcrawling zu Akzeptanzproblematiken der Telemedizin. doi:10.5281/zenodo.4557100.

Evelyn Gius, Svenja Guhr, and Benedikt Adelmann. 2020. d-Prose 1870–1920. doi:10.5281/zenodo.4315208.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Daniël de Kok and Sebastian Pütz. 2019. TüBa-D/DP Stylebook. Technical report, Seminar für Sprachwissenschaft, University of Tübingen.

Daniël de Kok and Tobias Pütz. 2020. Self-distillation for German and Dutch dependency parsing. In *Computational Linguistics in the Netherlands Journal*, volume 10, pages 91–107.

Jenny Kunz and Marco Kuhlmann. 2020. Classifier Probes May Just Learn from Linear Context Features. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5136–5146.

Arne Köhn. 2016. Evaluating Embeddings using Syntax-based Classification Tasks as a Proxy for Parser Performance. In *Proceedings of the 1st Workshop on Evaluating Vector Space Representations for NLP*, pages 67–71.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank Tensors for Scoring Dependency Structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1381–1391.

Alessio Miaschi, Dominique Brunato, Felice Dell'Orletta, and Giulia Venturi. 2020. Linguistic Profiling of a Neural Language Model. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 745–756.

Marius Mosbach, Stefania Degaetano-Ortlieb, Marie-Pauline Krielke, Badr Abdullah, and Dietrich Klakow. 2020. A Closer Look at Linguistic Knowledge in Masked Language Models: The Case of Relative Clauses in American English. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 771–787.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160.

Anthony Rios and Brandon Lwowski. 2020. An Empirical Study of the Downstream Reliability of Pre-Trained Word Embeddings. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3371–3388.

Anne Schiller, Simone Teufel, Christine Stöckert, and Christine Thielen. 1999. Guidelines für das Tagging deutscher Textcorpora mit STTS. Technical report, Institut für maschinelle Sprachverarbeitung, Seminar für Sprachwissenschaft, Stuttgart, Tübingen.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does String-Based Neural MT Learn Source Syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534.

Anders Søgaard, Anders Johannsen, Barbara Plank, Dirk Hovy, and Hector Martinez. 2014. What's in a *p*-value in NLP? In *Proceedings of the Eighteenth Conference on Computational Language Learning*, pages 1–10.

Ivan Titov and James Henderson. 2007. Fast and Robust Multilingual Dependency Parsing with a Generative Latent Variable Model. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 947–951.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 947–953.

Gözde Gül Şahin, Clara Vania, Ilia Kuznetsov, and Iryna Gurevych. 2020. LINSPECTOR: Multilingual Probing Tasks for Word Representations. *Computational Linguistics*, 46(2):335–385.

Appendix

text	overall tokens	sentences			
		count	token count		
			avg	m	stddev
dystopias	7,474	470	15.90	13	11.23
19th century	7,662	459	16.69	14	12.11
webcrawling	7,082	454	15.60	12	14.53
total	22,218	1,383	16.065	13	12.684

Table 5: Total number of tokens as well as sentence count and average, median and standard deviation of the number of tokens per sentence in our test sets

text	traditional			normal		empty		zero		cube		ccube		gauss		sphere	
	Parser	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
dystopias	Mate	0.88	0.82	0.92	0.89	0.89	0.84	0.89	0.84	0.89	0.84	0.91	0.87	0.90	0.86	0.91	0.87
19th century	Mate	0.85	0.80	0.89	0.86	0.87	0.81	0.87	0.81	0.87	0.82	0.88	0.84	0.88	0.83	0.88	0.83
webcrawling	Malt	0.85	0.80	0.90	0.87	0.87	0.83	0.86	0.82	0.87	0.83	0.89	0.85	0.88	0.85	0.89	0.85
all three	Mate	0.86	0.80	0.90	0.87	0.88	0.83	0.87	0.82	0.88	0.83	0.89	0.85	0.88	0.85	0.89	0.85

Table 6: attachment accuracies for the uninformative word embeddings (like Tab. 1), ignoring punctuation

text	traditional			normal		empty		zero		cube		ccube		gauss		sphere	
	Parser	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
dystopias	Mate	0.88	0.82	0.92	0.89	0.91	0.88	0.91	0.88	0.91	0.88	0.91	0.89	0.91	0.88	0.91	0.88
19th century	Mate	0.85	0.80	0.89	0.86	0.89	0.85	0.89	0.85	0.89	0.85	0.89	0.86	0.89	0.85	0.89	0.85
webcrawling	Malt	0.85	0.80	0.90	0.87	0.89	0.86	0.88	0.86	0.90	0.87	0.90	0.87	0.90	0.86	0.90	0.87
all three	Mate	0.86	0.80	0.90	0.87	0.89	0.87	0.89	0.87	0.90	0.87	0.90	0.87	0.90	0.86	0.90	0.87

Table 7: attachment accuracies for the uninformative POS embeddings (like Tab. 2), ignoring punctuation

text	traditional			normal		dystopias		19th century		webcrawling		total	
	Parser	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
dystopias	Mate	0.88	0.82	0.92	0.89	0.91	0.88	0.91	0.88	0.91	0.88	0.91	0.88
19th century	Mate	0.85	0.80	0.89	0.86	0.89	0.85	0.89	0.85	0.89	0.85	0.89	0.85
webcrawling	Malt	0.85	0.80	0.90	0.87	0.89	0.86	0.89	0.86	0.89	0.86	0.89	0.86
all three	Mate	0.86	0.80	0.90	0.87	0.90	0.86	0.90	0.86	0.90	0.86	0.90	0.86

Table 8: attachment accuracies for the domain-specific word embeddings (like Tab. 3), ignoring punctuation

text	empty		zero		cube		ccube		gauss		sphere	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
dystopias	***0.0050***	***0.0010***	***0.0200***	***0.0010***	6.5989	***0.0670***	27.6917	34.9537	20.3168	14.8269	25.7857	21.0308
19th century	*0.9970*	***0.1160***	*2.4010*	***0.1450***	29.1617	*3.3620*	49.0645	46.4145	43.1016	36.6626	47.5075	40.9466
webcrawling	*0.3660*	7.1739	*0.3280*	5.7079	26.5617	25.0447	44.1776	45.7915	42.6606	35.2796	45.0225	47.3175
all three	*1.2110*	*0.7660*	*1.8830*	*0.7500*	25.9947	8.0109	46.2675	48.6785	39.1796	33.5217	42.4446	39.6556

(a) *p*-values for the hypothesis that the results are not worse than Sticker's performance

text	empty		zero		cube		ccube		gauss		sphere	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
dystopias	20.1508	*0.8100*	8.9059	*0.8490*	***0.0380***	***0.0010***	***0.0010***	***0.0010***	***0.0020***	***0.0010***	***0.0010***	***0.0010***
19th century	*1.0150*	***0.0550***	*0.4340*	***0.0460***	***0.0030***	***0.0010***	***0.0010***	***0.0010***	***0.0010***	***0.0010***	***0.0010***	***0.0010***
webcrawling	***0.2430***	***0.0010***	*0.3280*	***0.0010***	***0.0010***	***0.0010***	***0.0010***	***0.0010***	***0.0010***	***0.0010***	***0.0010***	***0.0010***
all three	*4.1960*	***0.1660***	*2.5450*	***0.1890***	***0.0330***	***0.0050***	***0.0080***	***0.0010***	***0.0090***	***0.0010***	***0.0080***	***0.0010***

(b) *p*-values for the hypothesis that the results are not better than the performance of the respective best conventional parser (see Table 2)

Table 9: *p*-values (in %) for Yeh's randomized permutation test on performance differences between the uninformative POS embeddings and the two baselines. Values below the significance threshold of 5 % are marked in italics; values below the stricter threshold of 0.25 % are additionally marked in bold. Values for the combination of all three corpora were computed on a subset of 461 sentences so that *p*-values are comparable.

text	dystopias		19th century		webcrawling		total	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
dystopias	15.4958	12.1299	19.7578	13.7949	18.7668	6.8609	22.4608	10.3269
19th century	26.1267	20.1548	36.5816	33.2397	34.4367	14.9089	45.3085	26.7537
webcrawling	15.7938	17.4818	27.3167	31.9647	23.1158	21.8738	11.2789	12.4249
all three	25.5267	22.8208	33.2327	31.5717	31.0637	20.3748	30.1267	21.8918

(a) *p*-values for the hypothesis that the results are not worse than Sticker's performance

text	dystopias		19th century		webcrawling		total	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
dystopias	***0.0050***	***0.0010***	***0.0050***	***0.0010***	***0.0040***	***0.0010***	***0.0020***	***0.0010***
19th century	***0.0090***	***0.0010***	***0.0020***	***0.0010***	***0.0020***	***0.0010***	***0.0010***	***0.0010***
webcrawling	***0.0010***	***0.0010***	***0.0010***	***0.0010***	***0.0010***	***0.0010***	***0.0010***	***0.0010***
all three	***0.0350***	***0.0010***	***0.0150***	***0.0010***	***0.0160***	***0.0010***	***0.0210***	***0.0010***

(b) *p*-values for the hypothesis that the results are not better than the performance of the respective best conventional parser (see Table 3)

Table 10: *p*-values (in %) for Yeh's randomized permutation test on performance differences between the domain-specific embeddings and the two baselines. Values below the significance threshold of 5 % are marked in italics; values below the stricter threshold of 0.25 % are additionally marked in bold. Values for the combination of all three corpora were computed on a subset of 461 sentences so that *p*-values are comparable.

label	gold count	normal		empty		zero		cube		ccube		gauss		sphere	
		P	R	P	R	P	R	P	R	P	R	P	R		
APP	704	0.75	0.88	0.73	0.86	0.72	0.85	**0.62**	0.87	0.65	0.87	**0.62**	0.87	0.71	0.87
GMOD	384	0.95	0.96	**0.79**	**0.85**	**0.78**	**0.84**	0.88	0.90	0.90	0.91	0.89	0.91	0.91	0.90
KOM	110	0.89	0.72	0.88	0.68	0.89	0.70	0.87	0.69	0.88	**0.89**	0.87	**0.90**	0.86	0.73
NEB	224	0.84	0.83	0.76	**0.72**	0.81	**0.66**	0.79	0.78	0.86	0.79	0.88	0.81	0.82	0.79
OBJA	928	0.88	0.90	**0.70**	**0.77**	**0.69**	**0.76**	**0.73**	0.82	0.80	0.85	0.78	0.85	0.84	0.85
OBJD	163	0.78	0.77	**0.39**	**0.17**	**0.36**	**0.21**	**0.64**	**0.37**	**0.62**	**0.64**	**0.64**	**0.64**	**0.63**	0.67
OBJI	109	0.72	0.82	0.70	0.78	0.71	0.80	0.68	0.75	0.70	0.80	0.71	0.77	0.74	0.81
OBJP	114	0.51	0.28	**0.25**	**0.02**	0.50	**0.06**	**0.32**	**0.05**	**0.38**	0.24	**0.34**	0.18	0.41	0.22
PRED	277	0.83	0.85	**0.71**	**0.69**	0.73	**0.67**	**0.70**	**0.66**	0.81	0.75	0.77	**0.74**	0.77	0.77
ROOT	3466	1.00	0.99	1.00	0.99	1.00	0.99	1.00	**0.86**	1.00	**0.85**	1.00	**0.85**	1.00	0.95
S	1726	0.91	0.86	0.90	0.86	0.90	0.86	**0.78**	0.84	**0.76**	0.85	**0.79**	0.85	0.86	0.85

Table 11: Precision and recall for selected labels when parsing with the uninformative word embeddings. The 'gold count' column gives the number of occurrences of the label in our test data. Values differing by more than 10 percentage points from the baseline are marked in bold.

label	gold count	normal		empty		zero		cube		ccube		gauss		sphere	
		P	R	P	R	P	R	P	R	P	R	P	R		
APP	704	0.75	0.88	0.69	0.93	0.70	0.92	0.75	0.88	0.76	0.86	0.76	0.88	0.74	0.88
GMOD	384	0.95	0.96	0.92	0.96	0.95	0.96	0.96	0.96	0.95	0.95	0.95	0.95	0.95	0.95
KOM	110	0.89	0.72	0.89	**0.89**	0.90	**0.91**	0.88	0.68	0.89	0.68	0.87	0.67	0.86	0.68
NEB	224	0.84	0.83	0.84	0.85	0.86	0.86	0.83	0.80	0.84	0.81	0.85	0.82	0.81	0.80
OBJA	928	0.88	0.90	0.86	0.92	0.86	0.91	0.86	0.90	0.89	0.90	0.86	0.89	0.89	0.90
OBJD	163	0.78	0.77	0.81	0.76	0.80	0.80	0.76	0.79	0.75	0.83	0.75	0.79	0.77	0.82
OBJI	109	0.72	0.82	**0.90**	0.85	**0.91**	0.86	0.69	0.80	0.73	0.79	0.74	0.81	0.72	0.80
OBJP	114	0.51	0.28	0.47	0.25	0.52	0.30	0.46	0.28	0.45	0.29	0.42	0.24	0.46	0.29
PRED	277	0.83	0.85	0.79	0.83	0.79	0.82	0.84	0.84	0.80	0.84	0.83	0.86	0.80	0.83
ROOT	3466	1.00	0.99	1.00	**0.86**	1.00	**0.86**	1.00	0.91	1.00	0.99	1.00	0.99	1.00	0.99
S	1726	0.91	0.86	0.83	0.87	**0.80**	0.87	0.83	0.85	0.90	0.86	0.90	0.86	0.90	0.86

Table 12: Precision and recall for selected labels when parsing with the uninformative POS embeddings. The 'gold count' column gives the number of occurrences of the label in our test data. Values differing by more than 10 percentage points from the baseline are marked in bold.

label	gold count	normal		dystopias		19th century		webcrawling		total	
		P	R	P	R	P	R	P	R		
APP	704	0.75	0.88	0.73	0.88	0.74	0.89	0.73	0.88	0.73	0.88
GMOD	384	0.95	0.96	0.94	0.93	0.94	0.94	0.93	0.94	0.94	0.92
KOM	110	0.89	0.72	0.88	0.69	0.90	0.71	0.88	0.71	0.88	0.71
NEB	224	0.84	0.83	0.86	0.82	0.88	0.80	0.88	0.82	0.81	0.81
OBJA	928	0.88	0.90	0.87	0.88	0.86	0.89	0.85	0.87	0.85	0.88
OBJD	163	0.78	0.77	0.70	0.79	0.75	0.74	**0.67**	0.72	0.72	0.78
OBJI	109	0.72	0.82	0.69	0.82	0.69	0.82	0.74	0.81	0.72	0.82
OBJP	114	0.51	0.28	0.46	0.25	0.46	0.28	0.49	0.29	0.41	0.27
PRED	277	0.83	0.85	0.81	0.81	0.82	0.83	0.81	0.81	0.81	0.79
ROOT	3466	1.00	0.99	1.00	0.99	1.00	0.99	1.00	0.99	1.00	0.99
S	1726	0.91	0.86	0.91	0.86	0.91	0.86	0.91	0.86	0.91	0.85

Table 13: Precision and recall for selected labels when parsing with the domain-specific word embeddings. The 'gold count' column gives the number of occurrences of the label in our test data. Values differing by more than 10 percentage points from the baseline are marked in bold.