Chapter 7

Investigating the effect of automatic MWE recognition on CCG parsing

Miryam de Lhoneux
Uppsala University

Omri Abend
Hebrew University of Jerusalem

Mark Steedman
University of Edinburgh

We investigate the use of automatic Multiword Expressions (MWEs) recognition in parsing with Combinatory Categorial Grammar. We transform the representation of MWEs in CCGbank by collapsing them to one token. Our model significantly outperforms the baseline on the transformed gold standard showing the benefit of having this information at training time. It also performs significantly better on the transformed gold standard when the transformation is done before parsing as opposed to after parsing which shows that it can help the parser at prediction time. We conclude that despite the limited settings (our transformation algorithm is only able to deal with MWEs that do not cross constituent boundaries), our method can lead to improvements. We obtain different results with MWE recognisers that detect different types of MWE and therefore emphasize the need to experiment with different recognisers to find out which ones this method is best suited to.

1 Introduction

1.1 Motivation

Multiword Expressions (henceforth MWE(s)) are increasingly receiving attention in NLP. They represent a wide variety of phenomena with different properties but are generally agreed to be a group of multiple lexemes which have some

Miryam de Lhoneux, Omri Abend & Mark Steedman. 2019. Investigating the effect of automatic MWE recognition on CCG parsing. In Yannick Parmentier & Jakub Waszczuk (eds.), *Representation and parsing of multiword expressions: Current trends*, 183–215. Berlin: Language Science Press. DOI:10.5281/zenodo.2579045

level of idiomaticity or irregularity (Sag et al. 2002). They represent varied phenomena but, due to this irregularity, they are all generally considered a problem for NLP tasks and they are often a problem for syntactic parsing.

Recent research is showing that information about MWEs can help the syntactic parsing task (Nivre & Nilsson 2004; Korkontzelos & Manandhar 2010) and inversely, information about syntactic analysis helps MWE identification (Green et al. 2013; Weller & Heid 2010; Martens & Vandeghinste 2010). Working on either of the tasks by using information from the other has thus proven to be a useful thing to do and adding MWE information to the syntactic parsing task has proven useful in that it has helped increase parsing accuracy. Work on adding MWE information to syntactic parsing so far has been restricted to certain types of MWEs (multiword nouns, numerical expressions and compound function words in Nivre & Nilsson (2004), compound nominals, proper names and adjective-noun constructions in Korkontzelos & Manandhar (2010)) and hence leaves room for improvement.

Combinatory Categorial Grammar (henceforth CCG) is a strongly lexicalized formalism that is increasingly being used for parsing in NLP applications because of its computational and linguistic properties and because CCG parsers perform relatively well on the Penn Treebank (PTB). To give just a few examples, CCG is used in Machine Translation (e.g. Birch et al. 2007), sentence realization (e.g White 2006), semantic parsing and language acquisition (e.g. Krishnamurthy & Mitchell 2012), open-domain question answering and entailment (e.g. Lewis & Steedman 2013).

For these reasons, CCG parsing is an ideal framework to carry on the work on the interaction between syntactic parsing and MWEs and because CCG is a lexicalized formalism and thus encodes a lot of information in the lexicon, it would be useful to work on it by providing it with information about MWEs.

1.2 Aims

No work so far has tried to use MWE information to improve CCG parsing which is what we intend to do in this work. Different approaches to using MWE information for improving syntactic parsing have been conducted so far with different syntactic models. We conduct one of them which will be argued to be far from ideal but a necessary first step useful to build a sound baseline. The approach we pursue consists in altering training and test data, i.e. transforming the representation of MWEs so that they form one lexical item in them (and hence retokenize the sentence). We experiment with different MWE recognition methods so as to find out if the approach works better with certain types of MWEs than with others.

The two research questions we therefore try to answer are first whether or not we can improve CCG parsing with MWEs and second whether or not applying the same transformation approach to different types of MWEs can lead to different results.

1.3 Overview of the chapter

We give an overview of the background literature to further support our motivations and elaborate on the research questions in Section 2. We then explain and motivate the methodology we propose to use in order to answer the research questions in Section 3. We present our experiments and results in Section 4. We conclude from our study and propose avenues of research in Section 5.

2 Background

2.1 Multiword expressions

MWEs is an umbrella term that has been used to characterize a wide variety of phenomena. The most commonly acknowledged definition of this term since Sag et al. (2002) is that it is a group of multiple lexemes which have some level of idiomaticity or irregularity. The multiple lexemes in a MWE are called MWE units in the remainder of this chapter for convenience. This idiomaticity may be lexico-syntactic such as in the unusual coordination of a preposition and an adjective in *by and large*. It may be semantic such as in the idiom *kick the bucket* in which the meaning of the whole is not dividable into the meaning of the parts. It may be pragmatic such as in *good morning* which has a meaning attached to the situation in which it is said. Finally, it may be statistical such as the collocation *strong coffee* in which both units occur more frequently than expected.

Different MWE types present different properties. They vary in flexibility: words may appear between the units of a flexible collocation (*strong home-made coffee*, for example) but not between the units of a lexically fixed figurative expression such as *it's raining cats and dogs*. They also vary in compositionality: *Strong coffee* is fully compositional whereas *kick the bucket* is not and *spill the beans* is semantically decomposable, i.e. the meaning of the whole is not predictable from the meaning of the parts but can be decomposed into its parts: if *spill* is interpreted as *reveal* and *the beans* as *the secret* (Nunberg et al. 1994). Despite these varied properties they are all generally agreed to be hard to deal with in NLP applications (Sag et al. 2002) and the importance of dealing with them properly has been increasing over the past decade. As described at length in Kim's (2008) thesis, "dealing with" MWEs consists in developing systems and models for various kinds of tasks. For syntactic analysis, it is important to identify them in text and extract them to a dictionary. For semantic understanding, it is important to measure their compositionality, classify and interpret them.

2.2 Combinatory Categorial Grammar

2.2.1 Presentation

Combinatory Categorial Grammar (Steedman 2000) is a strongly lexicalized grammar formalism which is currently gaining popularity in the NLP community.

CCG was built with the intent of being linguistically aware as well as computationally tractable partly as a reaction to transformationalist ideas which were predominant in formal grammars at the time. It differs from the latter mainly in having one component including syntactic and semantic information instead of having separate modules for each in the grammar. Similarly, instead of having a large amount of rules and a lexicon as is the case in traditional grammars, it has a small set of universal rules and a lexicon which encodes most syntactic information. For the sentence *John buys shares*, a traditional grammar has information in the lexicon: that *John* is an NP, that *buys* is a verb, that *shares* is an NP, and in the grammar: that a V and an NP form a VP and that an NP and a VP form a sentence S, as in Figure 1. By contrast, for the same sentence, CCG has information in its lexicon that *John* is an NP, that *shares* is an NP and that *buys* first takes an NP to its right then an NP to its left to form a sentence S, as in Figure 2.

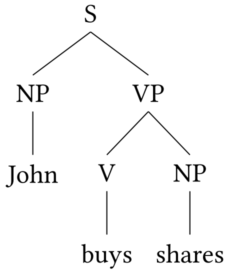

Figure 1: PTB-style tree

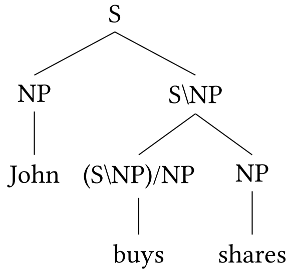

Figure 2: CCG tree

Without going into too much detail about how this works, lexical categories work either as functor or as argument and a set of combinatory rules allow them to combine. For example, the category (S\NP)/NP works as a functor that takes an NP to the right (indicated by the forward slash followed by an NP). The category of *buys* therefore can combine with the category of *shares* to result in the category S\NP which in turns takes an NP argument to the left (indicated by the backslash followed by an NP), which it finds in the category of *John* to form a sentence S.

This grammar architecture allows CCG to deal elegantly with long-range dependencies. Instead of adding a level of representation in the form of a trace as in Figure 3, the grammar has universal rules which allow the combination of lexical items, as shown in Figure 4. This has computational advantages and linguistic plausibility: linguistics is increasingly adopting a view of grammar where syntax and the lexicon are two modules that are not completely separate in the grammar but instead interact with each other. It is a tenet of the recently emerging framework of Construction Grammar (Hoffmann & Trousdale 2013). These linguistic and computational properties have made it a widely used framework across NLP research.

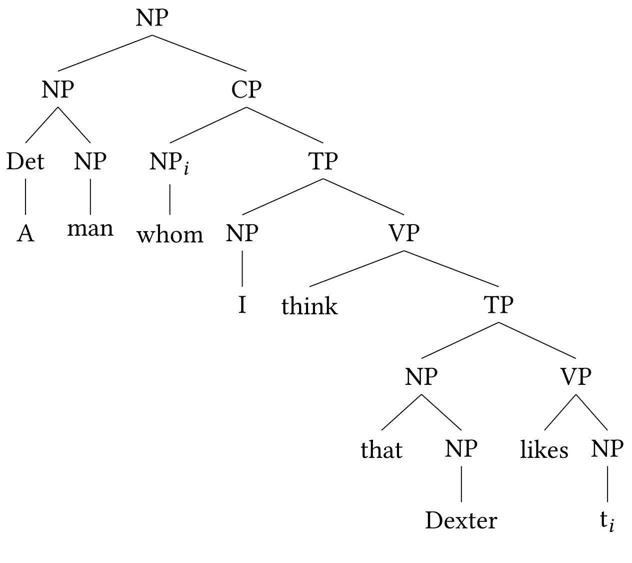

Figure 3: PTB-style tree

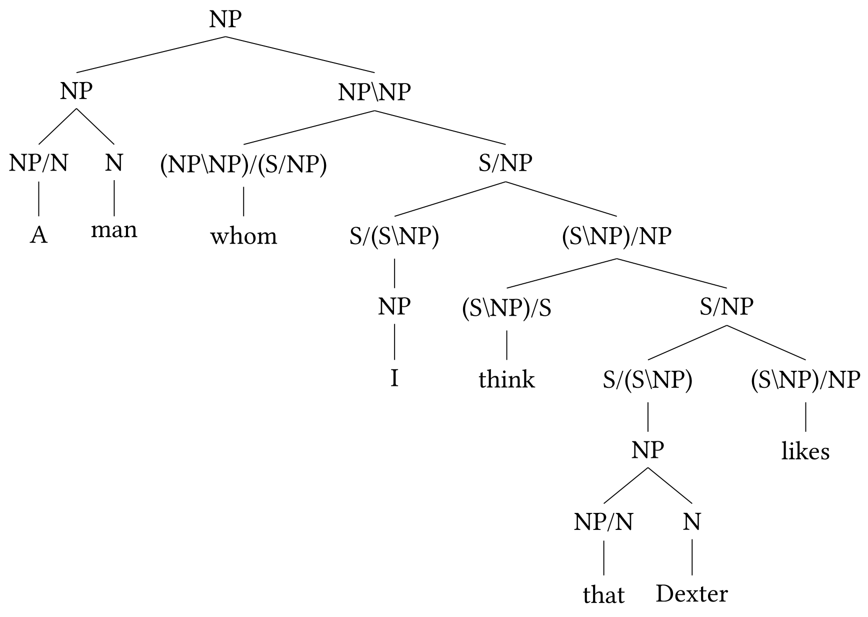

Figure 4: CCG tree

2.2.2 CCG parsing

The first efficient statistical model was the generative model built by Hockenmaier & Steedman (2002) and extended to a discriminative model by Clark & Curran (2007). Both use CCGbank (Hockenmaier & Steedman 2007), a CCG converted version of the PTB. Both models perform close to state-of-the-art although with simpler statistical models which is argued by the authors to be the result of having a more expressive grammar than the PCFGs used by state-of-the-art parsers.

2.3 Syntactic parsing and multiword expressions

As mentioned in Section 2.1, the identification of MWEs is important for syntactic analysis. Because they have unusual properties, however, their analysis can be quite problematic. The question of how to deal with MWEs for syntactic parsing has been raised by many researchers. It has been approached in different ways. Researchers working with precision grammars such as HPSG for example have accommodated the lexical entries for MWEs in the lexicon so that MWEs are not a problem for parsing. Researchers on data-induced grammars have accommodated the testing and/or training data before parsing. Recent research has proposed to both change the lexicon and the parsing algorithm. A last recent approach is to learn MWE representations and dependency trees jointly. We briefly describe each of these approaches in turn. We describe the second approach in more details than the other two because this is the one we use in this work for reasons explained in Section 3.1.

2.3.1 Transforming the lexicon

Different types of lexical entries have been proposed for MWEs in the grammar. A lot of research proposes to simply analyse all MWEs as "words-with-spaces", i.e. group the MWE units together in the syntactic analysis. This analysis has been argued against by many. Sag et al. (2002) have suggested sophisticated ways of representing the different MWE types in a grammar, which have been partly implemented within the framework of the precision grammar HPSG, as described by Copestake et al. (2002). Zhang et al. (2006) established that MWEs are a tremendous source of parse failures when parsing with a precision grammar such as HPSG and henceforth proposed a way of using this information to identify new MWEs and enrich a lexicon: they suggested using parse failures to predict the existence of a MWE.

2.3.2 Transforming the data

Since the seminal work of Nivre & Nilsson (2004), research has shown that treating MWEs as one token or a "word-with-spaces" in test and/or in training data before parsing and/or training leads to an improvement in parsing accuracy. Nivre & Nilsson (2004) have shown that to be true for deterministic dependency parsing and Korkontzelos & Manandhar (2010) have shown that to be true for shallow parsing.

The approaches adopted in these two papers are quite different and we describe each in turn.

2.3.2.1 Transforming training and test data

Nivre & Nilsson (2004) created two versions of a treebank, one in which MWEs are annotated as if compositional and one in which they are joined as one lexical item. They show that training a parser on the second version of the treebank leads to a better parsing accuracy. They use a corpus with manual MWE annotation to create both versions of the treebank and hence simulate "perfect" MWE recognition. MWE annotation, however, only consists in a few MWE types so it is not comprehensive. They report improvement in parsing accuracy of the MWEs themselves but also of their surrounding syntactic structure. They opened the gate for improving syntactic parsing with MWE information but left many questions unanswered. For example, the question of whether or not their results port to other syntactic parsing models, whether or not the full potential they obtained with "perfect" recognition of MWEs can be obtained with an automatic recogniser and whether or not this potential can be increased when recognising other types of MWEs. Some of these questions have been partially addressed since then. Constant et al. (2012) have shown that with an automatic recogniser, the parsing accuracy improvement is not as dramatic as predicted by Nivre & Nilsson (2004). Eryiğit et al. (2011) found out that in the case of a morphologically rich language (e.g., Turkish), the approach works with some types of MWEs but not with others.

2.3.2.2 Transforming test data

Korkontzelos & Manandhar (2010) reported similar parsing accuracy improvements for shallow parsing, showing that Nivre & Nilsson (2004)'s results do seem to port to at least one other parsing model. Their technique is, however, quite different. They created a corpus containing a large number of pre-selected MWEs (randomly chosen from WordNet) and converted it to a version in which the MWE units are collapsed to one lexical item. They POS-tag the two versions of the corpus before parsing each. They subsequently analyse the differences in output. In order to do so, they randomly select a sample of output from both parsed corpora and build a taxonomy of changes they observe from one to the other. For each class in the taxonomy, they determine whether the change in output led to increased accuracy, decreased accuracy or did not change the accuracy. They automatically classify the rest of the output data and observe an overall increase in accuracy. Their work not only confirms the results obtained from previous work but also provides an insightful qualitative analysis of changes obtained with their method. They believe the improvement in accuracy is partly due to the fact that the parsing model backs off to POS-tags for rare and unseen words. When MWE units are collapsed to one token, that token is not known by the parser but it still gets assigned a sensible POS-tag because the POS-tagger uses contextual information.

2.3.3 Transforming the lexicon and the parsing algorithm

A lot of work has shown that although MWE information improves syntactic parsing, the reverse is also true: syntactic analysis improves MWE identification.

Green et al. (2013) successfully tuned a parser for MWE identification, Weller & Heid (2010) and Martens & Vandeghinste (2010) showed that using parsed corpora for MWE identification is beneficial. These findings led Seretan (2013) to propose that neither accommodating the grammar with MWE information, nor recognising MWEs in raw text as a help to parsing are appropriate ways of dealing with the issue of MWEs in syntactic parsing because neither approach takes advantage of the fact that MWE information and syntactic analysis are mutually informative. She proposes instead to have a MWE lexicon and to deal with potential MWEs during parsing.

2.3.4 Joint learning of MWE identification and parsing

Based on the same observation that the tasks of MWE identification and parsing can inform each other, Constant & Nivre (2016) propose to learn both jointly. They use corpora that both have dependency and MWE annotations and modify the parsing algorithm so as to learn both representations jointly. They show that this approach is effective.

2.3.5 Advantages and caveats of the different approaches

All of these researchers have shown the importance of MWEs for syntactic parsing but all of the approaches presented have caveats. Research on HPSG seems to have found the most sophisticated methods of dealing with MWEs but parsing with precision grammars is known to be much less robust (Zhang & Kordoni 2008) than parsing with data-induced grammars which make it a suboptimal solution for practical parsing. Learning MWE representations and syntactic parsing jointly is probably the most promising approach but it requires a lot of manual work since it requires a corpus that is annotated both with MWEs and dependency trees. As far as other solutions are concerned, they are often very much limited by the type of MWEs that have been dealt with. All other solutions presented as a matter of fact concentrate on a few types of MWEs. However, as argued by Kim (2008), because of the different but interrelated properties of MWEs, it is neither appropriate to try and generalize from MWEs and find a single representation which works for all types, nor is it appropriate to deal with each MWE type at a time. An approach for improving syntactic parsing on all MWE types is still lacking and previous approaches leave the question of whether the results can be reproduced with different types of MWEs unanswered.

2.4 Research questions and objectives

In Section 2.3, it was said that MWE identification information improves syntactic parsing although current approaches to doing so leave room for improvement. Trying to improve syntactic parsing with MWE information therefore looks like a promising avenue of research. In Section 2.2, arguments for working with CCG parsing were put forward.

Very little attention has, however, been given to MWEs in CCG parsing. Constable & Curran (2009) modified CCGbank to have a better representation of verb-particle constructions but did not report any parsing accuracy improvement. No work has tried to establish whether CCG parsing accuracy could be improved by adding information about MWEs which is what we intend to do in this work. Our aim is twofold: we want to find out whether or not MWE information can improve CCG parsing and we wish to find out if using methods that have been used for a restricted set of types of MWEs can be extended to different types of MWEs.

3 Methodology

3.1 Approach

As explained in the last section, in this work we concentrate on an approach that consists in transforming the representation of MWEs in treebanks. In Section 2.3.2, two different versions of this kind of approach have been described. In the first, manual annotation is used to create two versions of the treebank. This left questions unanswered, two of which being whether or not the approach can work with automatic recognition and whether or not the approach can work with different types of MWEs. In this work, we conduct the type of approach described by Nivre & Nilsson (2004) using an automatic recogniser to answer the first of these questions in the context of CCG parsing. We also experiment with the recogniser by using different versions of it to answer the second of these two questions. This approach is especially interesting in that, as has been shown in Schneider et al. (2014) who attempted a comprehensive annotation of MWEs in a corpus, even manual annotation of MWEs is a difficult task and experimenting with different MWE recognisers could lead to interesting results.

Our approach therefore involves transforming both training and test data. Transforming the training data can help the parsing model learn more sensible representations of language. For example in the tree for *part of speech*, *of speech* is considered a modifier of *part* as in Figure 5 which does not make much sense. Instead, grouping the three lexical items as in Figure 6 gives a better representation of this group of words.

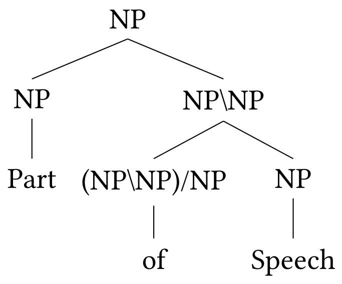

Figure 5: Traditional tree for *Part of Speech*

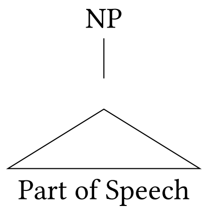

Figure 6: Tree for *Part of Speech* where tokens are grouped

Transforming the test data by for example collapsing the three lexical items *part*, *of* and *speech* to one token *part+of+speech* can help the parser make sensible decisions locally by telling it to consider the three words as one. For example, if this token is followed by a coordinator, the parser knows that coordinating one of the units is not a possibility. In the sentence *it gives part+of+speech and lemma information*, the parser cannot coordinate *speech* with *lemma* which would be a possibility otherwise. Transforming MWEs in training and test data leads to two different effects of adding MWE information to the syntactic parsing pipeline and it is best if we can differentiate both in the experiments. We call the first type of effect *training effect* and the second *parsing effect*. We repeat the definition of these two effects below.

Training effect: the parser learns more sensible representations of MWEs and its units.

Parsing effect: the parser is helped locally in its decision by considering the MWE units as one unit.

3.2 Parsing model

As mentioned in Section 2.2, both generative and discriminative models exist for parsing with CCG. There are different generative models with different properties. We chose to use StatOpenCCG, developed by Christodoulopoulos (2008) and recently further expanded by Deoskar et al. (2014) because of its ease of use, flexibility and fast training. The expansion of the parser by Deoskar et al. (2014) is particularly well suited to our purposes: it was extended so that it works better on unknown lexical items. Joining lexical items to one token will increase the sparsity of the data and being able to deal with unknown data is therefore a concern for our approach. More particularly, the model proposed by Christodoulopoulos (2008) and Deoskar et al. (2014) is based on one of Hockenmaier (2003)'s models called LexCat which conditions probabilities on lexical categories. Deoskar et al. (2014) make use of this LexCat model instead of the fully lexicalized model which conditions it on words precisely so that the parser is better equipped to deal with unseen lexical items. They introduce a smoothed lexicon to deal with these. They POS-tag the test data in a pre-processing stage and use POS-information to determine the lexical categories of words by using probabilities of lexical categories that appear with each POS-tag of unseen word in the seen data. Because, as mentioned in Section 2.3.2.2, Korkontzelos & Manandhar (2010) have shown that POS-tags assigned automatically to MWEs were useful when parsing, the LexCat model therefore looks ideal for our purposes. We follow Deoskar et al. (2014) in using the C&C tools (Curran et al. 2007) to POS-tag our test data so as to have a model that is comparable with theirs.

3.3 Extending the parsing model with MWE information

As explained in Section 3.1, the objective is first to recognise MWEs in the unlabeled version of CCGbank and then to collapse MWEs to one lexical item in the annotated version of the treebank and in the unlabeled test data. The MWE recognition part is described in Section 3.3.1 and the CCGbank conversion is described in Section 3.3.2.

3.3.1 Recognising MWEs

For MWE recognition, we use a tool developed by Finlayson & Kulkarni (2011). It can be used to build an index of MWEs with information about their probability. It can also be used with a default index which contains all the MWEs and inflections extracted from Wordnet 3.0 and Semcor 1.6 and statistics for each MWE. There are three different tools of interest to us. Simple detectors detect MWEs in text. There is a detector to find proper nouns, one to find all types of MWEs that are in the index, one that finds MWEs that contain only stop words, etc. These simple detectors can also be combined to form a complex detector. There are filters which filter the results of detectors. One for example only accepts MWEs that are continuous, one throws out MWEs which have a score under a certain threshold, one only keeps MWEs under a certain length. The last tool we need is called a RESOLVER and it resolves conflicts when lexical items are assigned to more than one MWE. Conflicts can be resolved in different ways: one resolver picks the leftmost MWE. For example, let us say we have an input sentence that includes *new york life insurance.* If the MWE index contains *new york life* and *life insurance*, the resolver will return *new york life* but will not consider *insurance* as part of a MWE. Another resolver picks the longest matching MWE. For example, let us say we have an input sentence which contains *new york stock exchange.* If the MWE index contains *stock exchange*, *new york* but also *new york stock exchange*, the longest matching resolver will return *new york stock exchange* as a match.

Let us take the following sentence as an example of input for a resolver:

(1) Mr. Spoon said the plan is not an attempt to shore up a decline in ad pages in the first nine months of 1989; Newsweek's ad pages totaled 1,620, a drop of 3.2 % from last year, according to Publishers Information Bureau.

The resolver returns a list of its MWEs from left to right. In the case of our example (1), the output for example looks like this:

(2) mr._spoon, shore_up, according_to, publishers_information_bureau

The presented protocol can work with any type of MWE recogniser, provided that it is filtered to output only continuous MWEs and resolved so that any word can only appear in one MWE. This library therefore serves our purposes perfectly since it leaves quite a lot of room for experiments. Experiments are described in Section 4.

3.3.2 Transforming the treebank

The algorithm collapses the MWE units to one node when they form a constituent in the tree. The label of the node is the label of that constituent. For example, given the MWE *publishers_information_bureau* and the subtree in Figure 7, the algorithm returns the subtree in Figure 8.

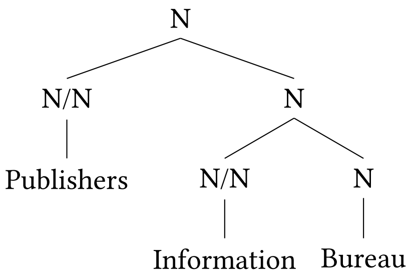

Figure 7: Original subtree

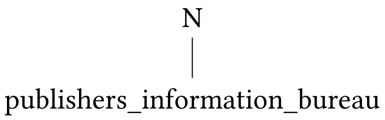

Figure 8: Transformed subtree

The algorithm discards MWEs if they do not form a constituent in the tree. An example of tree in which MWE units (e.g., *according to*) are not siblings in the tree is given in Figure 9. The ideal way in which it should be transformed is given in Figure 10 but attempting to find an algorithm which would work for all non-sibling cases is beyond the scope of this work. We tried our algorithm with a good recogniser, collected statistics and found that 79.5% of the cases (42,309/53,208) were siblings in the tree which we considered a good basis for experimentation. Note, however, that modifying those non-sibling MWEs would make bigger changes to the tree as it would not only remove the lexical categories of MWE units but it would additionally remove the parent category of the MWE units involved and create a new category for the whole MWE. As we will see in Section 4, dealing only with sibling MWEs leads to slight improvements, we hypothesize that an improved algorithm that can deal with non-sibling MWEs can lead to more substantial improvements.

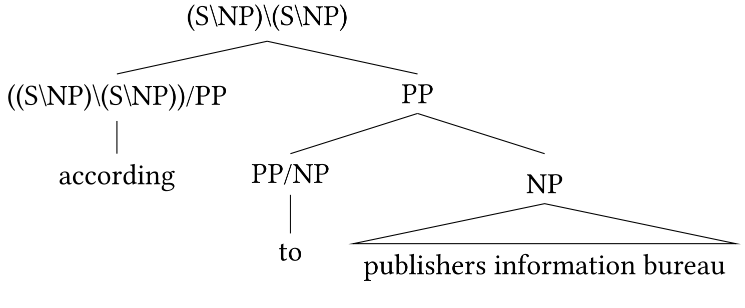

Figure 9: Tree with MWE units that are not siblings

Because, as explained in Section 3.5, we evaluate our method on dependency trees (which can be read off CCG trees), we also need to modify the gold standard dependency trees. Transforming dependency trees involves merging nodes

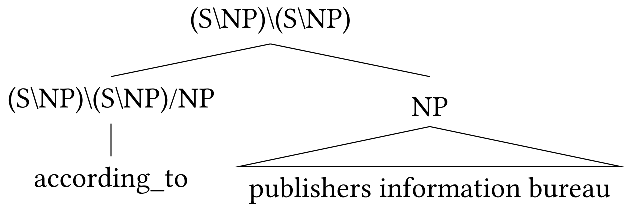

Figure 10: Ideal MWE non-sibling transformation

in the graph and changing edges according to the new nodes. When the nodes are merged, edges from the original dependency graph fall into three different categories. Let us take the dependency graph in Figure 11 as an example in which *Mr. Vinken* is a MWE. There are edges between two units of a MWE such as the one between *Mr.* and *Vinken.* We call these INTERNAL EDGES for convenience. Our algorithm removes them as shown in Figure 12. There are edges between a MWE unit and another word in the sentence such as the edge between *Vinken* and *is.* We call this type of edge a MEDIATING EDGE. In the transformed graph, the whole MWE becomes the node of that incoming or outgoing edge. The edge between *is* and *chairman* does not connect any MWE and does not need changing. We call it EXTERNAL EDGE.

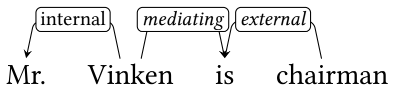

Figure 11: Dependency graph

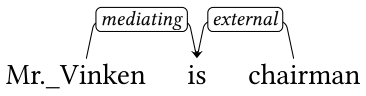

Figure 12: Transformed dependency graph

The downside of this algorithm is that it can create cyclic dependencies between lexical items, i.e. two nodes are connected by two edges going in the opposite direction. We tested the algorithm on the CCGbank and found that in practice this is not a major issue: only 7 cyclic dependencies were created in the ~48,000 sentences.

3.4 Training and parsing

We follow the tradition and use sections 01–22 of CCGbank for training, section 00 for development and section 23 for testing.

Using the same parameters as in Deoskar et al. (2014) to train and parse the test data, we obtain around 87% of correct lexical categories, which is similar to the result they report. This model serves as our baseline and we henceforth call it *model*$_A$.

For each experiment, we run the MWE recogniser on an unlabeled version of the CCGbank[1]. We then apply our cascaded algorithms described in the previous section to every sentence from the CCGbank. We train the model and parse the test file. We call the transformed treebank CCGbank$_B$ and the model trained on it *model*$_B$.

3.5 Evaluation

The traditional parsing accuracy metric PARSEVAL has been argued (for example by Clark & Hockenmaier (2002)) to be too harsh on CCG derivation trees because they are always binary, as opposed to PTB-style trees which can have flat constructions with more than one branching node. This binary nature of CCG trees make them prone to having more errors. Consequently evaluation of dependencies has generally been preferred for CCG parsing. As further argued by Clark & Hockenmaier (2002), it also makes sense to use dependencies to evaluate CCG parsing since one of the advantages of CCG over other formalisms is precisely its treatment of long-range dependencies.

In order to evaluate our models, we can thus extract dependencies from the parsed files and compare them with the gold standard. Because we changed the gold standard as compared to *model*$_A$ (as defined in the previous subsection), however, the results obtained from comparing our parsed files with our gold standard are not directly comparable with the results obtained when applying the same evaluation scheme to *model*$_A$. Therefore, we cannot directly compare *model*$_A$ with our *model*$_B$s (as defined in the previous subsection) which is essential in answering our research questions. Instead, we have to transform the data of one of the models so as to compare each of the models with the same gold standard. Because we assume that we have created a sensible gold standard with our transformation algorithm, we mainly use gold standard$_B$ for evaluation.

As mentioned in Section 3.1, transforming training and test data can lead to two different effects which can lead to an improved parsing accuracy, i.e. training (the parser learns useful information during training) and parsing effects (the trained parser is helped in its decisions by MWE information) which we would like to differentiate. This can be achieved by conducting different experiments with our existing models. We can assess training effect by testing whether or not *model*$_B$ can outperform *model*$_A$ when evaluated on the same gold standard.

[1]We created an unlabeled version of CCGbank from the tree leaves to make sure the data is compatible with the trees we work with when transforming trees.

We can assess parsing effect by testing whether or not *model*$_A$ can outperform itself when given transformed test data. We discuss these evaluation schemes in Section 3.5.1.

If there is training and/or parsing effect, we can assume that automatic recognition of MWEs can be used to improve syntactic parsing. We can verify this by testing whether or not we can use information from *model*$_B$ to outperform *model*$_A$ on gold standard$_A$. We use a second evaluation scheme where we combine information from output from *model*$_A$ and output from *model*$_B$ (henceforth called MODEL COMBINATION) to test this which we discuss in Section 3.5.2.

As mentioned in Section 2.4, we not only want to know whether or not information about MWEs can help CCG parsing but we are also interested in finding out whether or not different types of MWEs impact parsing accuracy in different ways. As will be explained in Section 4, we created different versions of CCGbank$_B$ and different versions of *model*$_B$ with these. Because we created different gold standard for each of these models, they cannot directly be compared. Instead, comparing how different *model*$_B$s can improve *model*$_A$ is possible by comparing their combination with it against gold standard$_A$. Again then we can use model combination and combine information from output from *model*$_A$ with information from output from *model*$_B$. We compare this combined model output against gold standard$_A$ and compare the results when combining *model*$_A$ with different versions of *model*$_B$. We discuss how this can be achieved in Section 3.5.3.

3.5.1 Assessing training and parsing effects

Modifying the output from *model*$_A$ so that it is comparable with the gold standard from *model*$_B$ is straightforward: we just need to apply the transformation algorithms to the output from *model*$_A$ with the MWEs found in the test data. We can also test *model*$_A$ on data transformed before parsing.

3.5.1.1 Parsing effect

Testing whether there is a parsing effect can be done by testing whether or not *model*$_A$ can perform better on test data transformed before parsing than on test data transformed after parsing. We conduct this evaluation. There is a caveat in this evaluation, however: we are using information from the gold standard in the test data, i.e. we know which MWEs are siblings in the test data. This introduces an artefact which makes the results somewhat difficult to interpret: the transforming before parsing method has sibling information which the transforming after parsing method does not. There can be parsing and sibling effects and the two cannot be decoupled. A way to circumvent this problem is to transform MWEs regardless of their sibling status (i.e. treat all detected MWEs as if they were siblings) and compare the model when we transform before parsing with the model when we transform after parsing. Transforming all MWEs in unlabeled test data is straightforward. As mentioned in Section 3.3.2, we do not have an algorithm to transform MWEs that are not siblings in trees so we cannot transform the output parse trees. However, since we are working only with dependencies for evaluation, it is possible to transform all MWEs in all the dependencies of the sentence. The problem with this evaluation is that the output cannot perform well on gold standard$_B$ because it is not tokenized in the same way and we treat dependencies wrongly tokenized as errors. However, both transforming before and transforming after parsing suffer from the same problem and the comparison between the two is fair.

3.5.1.2 Training effect

Testing whether there is a training effect consists in comparing the results of *model$_A$* on transformed data with the results of *model$_B$* on transformed data. In this evaluation, the caveat that we are using information from the gold standard in the test data can also be considered problematic because *model$_B$* is trained on data with information about siblings. This information is unseen by *model$_A$*. We therefore test both models on data where only siblings are transformed (called GOLD TEST for convenience) and on data where all MWEs are transformed (called FULLY TRANSFORMED TEST for convenience). Again, the problem with this evaluation is that the output cannot perform well on gold standard$_B$ because it is not tokenized in the same way. Again, however, both models suffer from the same problem and the comparison between the two is fair.

3.5.2 Verifying whether or not automatic recognition of MWEs can improve CCG parsing on the original gold standard

Results which will be discussed in Section 4 seem to indicate that there is both a training and a parsing effect and that *model$_B$* performs better than *model$_A$* on some dependencies. Our findings support the claim that automatic recognition of MWEs can improve CCG parsing. These results, however, led us to want to verify whether or not *model$_B$* can improve the score on the standard evaluation benchmark, i.e. on gold standard A. This involves “detransforming” the output from *model$_B$* and splitting MWEs back into their units. However, by transforming the data, we have lost information about some dependencies in the sentence.

We have no *internal edges* (edges between MWE units of the same MWE) and when there is a *mediating edge* (edges between MWE units of any MWE and other words in the sentence) we do not know which MWE unit of the MWE should the incoming or outgoing node of that edge. In Figure 12 reproduced in Figure 13 for convenience, we do not know whether the label between *is* and *mr._-vinken* should come from *mr* or *vinken*. For this reason, we propose to combine information obtained from parsing the test data with our transformed model $model_B$ with information obtained from parsing the test data with the original model $model_A$. We therefore take some dependencies from output$_A$ and some from output$_B$.

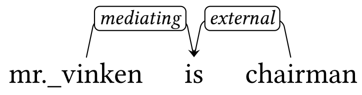

Figure 13: Transformed dependency graph

External edges can be taken from output$_B$. *Internal edges* do not exist in output$_B$. Hence we propose to take them from output$_A$. For *mediating edges*, there are different possibilities. We can take them from output$_A$ and therefore only test whether or not $model_B$ performs better than $model_A$ on *external edges*. We call this combination method *medFromA*. If we want to test $model_B$ on *mediating edges*, we can take *mediating edges* from output$_B$. For this to work, the model combining algorithm needs to choose one node as the incoming or outgoing node of that edge: in our example, it should either be *Mr.* or *Vinken*. We use two additional combination methods. We use one in which the rightmost node is chosen as incoming or outgoing node for *mediating edges* from output$_B$ which we call the *rightmostMed* scheme. We also use one in which the leftmost node is chosen which we call the *leftmostMed* scheme. In order for the model combining algorithm to work, we need to recover information about MWEs and their units and hence to know for each dependency if we are dealing with an *internal*, *external* or *mediating edge*. This can easily be done because MWE and their units are annotated in the unlabeled data.[2]

When these models are combined in these three different ways, we have a new combined model that we can compare with $model_A$ on gold standard$_A$. In this case, using "gold test" data is again problematic. As a matter of fact, if we use output$_B$ as obtained after parsing "gold test" data, we are using information obtained during the conversion of the gold standard and we are using a parsing

[2]Our MWE recogniser joins MWE units of a MWE by a "+" symbol.

pipeline which is not fully automatic. In order to make sure that we can outperform *model*$_A$ in a fully automatic way, we can use parses of *model*$_B$ tested on the "fully transformed test" data set as described in Section 3.5.1, and combine them in the same three ways as described above.

3.5.3 Testing whether or not different MWE types impact the results differently

As mentioned before, we use different MWE recognisers to create different versions of CCGbank$_B$ and hence different versions of *model*$_B$. Because we created a different gold standard for each, results from different models are not directly comparable. We can, however, convert output parses using the model combination algorithm described in Section 3.5.1 and test each model against gold standard$_A$. In this way, different versions of *model*$_B$ can be compared.

3.6 Summary of the experimental setup

Figures 14 and 15 summarize our experimental setup.

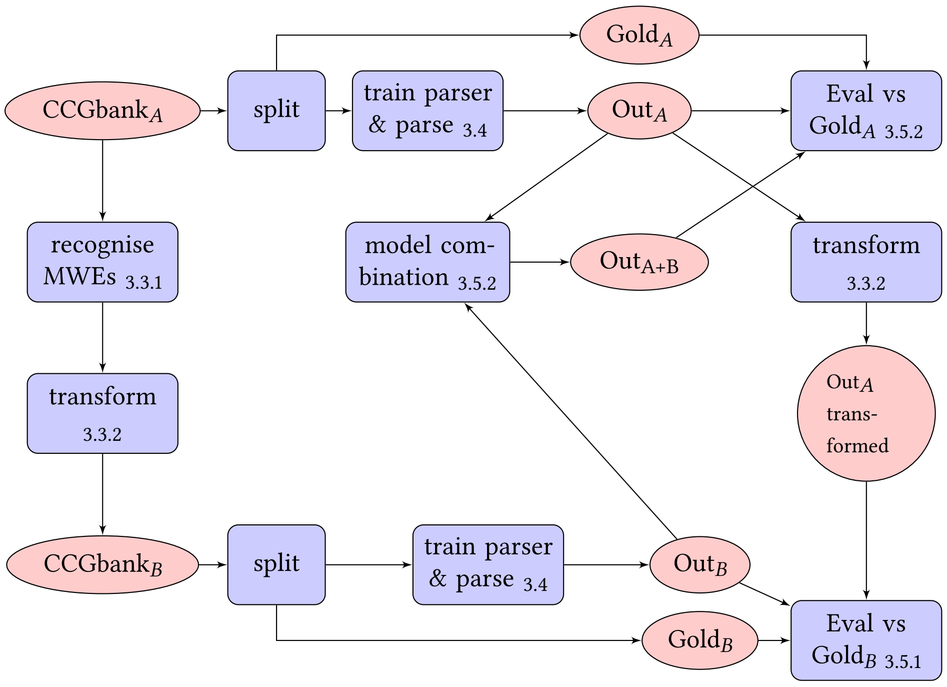

Figure 14: Pipeline of an experiment on one version of one application of MWE recognition to the parsing pipeline with all the evaluation schemes that can be applied to it. The transforming before parsing of *model*$_A$ (see Section 3.5.1) is omitted for clarity and given in Figure 15.

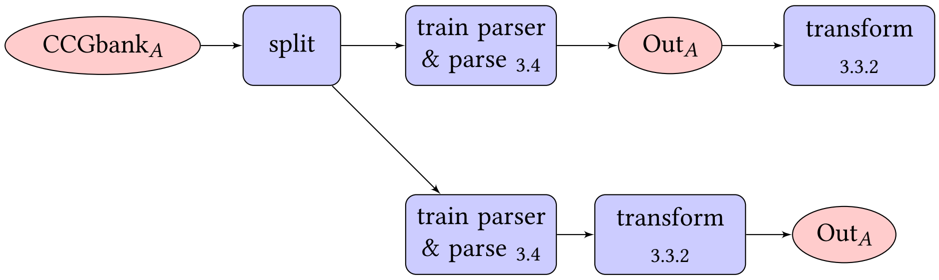

Figure 15: Pipeline of the "transforming before parsing" against the "transforming after parsing experiment."

In Section 2, we motivated our study and identified two research questions: whether or not information about MWEs can improve CCG parsing and whether or not different types of MWEs can influence parsing accuracy in different ways. In this Section, we proposed a methodology for testing this. We refined the first research questions: what we want to find out is whether or not automatic recognition of MWEs can improve CCG parsing. Additionally, we separated it into two further research questions: whether we can observe a parsing effect (the parser is helped in its decisions by transformed data) and/or whether we can observe a training effect (the parser learns something useful). We proposed to use different MWE recognisers to answer the second question. When defining an algorithm for transforming the treebank, however, we could not find a straightforward algorithm to transform MWEs that are not siblings in the tree and decided to settle for an algorithm that only transforms siblings. This led to further complications in the evaluation schemes because it makes it harder to give a fair evaluation of our models. We found ways to circumvent the problems: we proposed different evaluation schemes together with cross-validations. We now turn to the results of our experiments.

4 Results

In this section, we look at each of the research questions in turn. To assess statistical significance of our best results, we use a one-tailed randomized shuffling test with 10,000 iterations. We use the software created by Padó (2006) (slightly modified in order to make it a one-tailed test instead of a two-tailed one) for our tests.

4.1 MWE recognition

We use the jMWE library described in Section 3.3.1 with the default index which contains MWEs from Wordnet 3.0 and Semcor 1.6. We use the library's three different tools which were explained in that section. Those tools are detectors which detect MWEs in text, filters which filter through the results of one or more detectors and resolvers which resolve conflicts between MWEs when one word is assigned to more than one MWEs by the detector.

We use the following tools:

- Detectors:
  - Proper Nouns: detects proper nouns, like *wall street.*
  - Stop words: detects MWEs that only contain stop words, like *instead of.*
  - Exhaustive: finds all MWEs that are in the index.
- Filters:
  - MoreFreqAsMWE: only keeps MWEs if its units appear more often together than apart in the corpora in which they were collected.
  - ConstrainLength: only keeps MWEs that have 2 units.
- Resolvers:
  - Longest: always picks the longest matching MWEs.
  - Leftmost: picks the MWE that starts earliest in the sentence.

We build 5 different MWE recognisers with different combinations of these tools. This means that the study is by no means exhaustive. Information about our recognisers and statistics about the MWEs they detect are summarized in Table 1. The numbers in column "ID" denote the recognisers used in the remainder of this section. Similarly, each $model_B$ is denoted by the recogniser which was used to train it as indicated by this number.

4.2 Can we improve CCG parsing accuracy with automatic MWE recognition?

As explained in Section 3.5, we use different evaluation schemes to answer this question. First we evaluate $model_B$ and $model_A$ against gold standard$_B$ and determine whether there is training and/or parsing effects. Then we verify whether

Table 1: Description (detector, filter and resolver) of MWE recognisers used and statistics of MWEs collected with them in the treebank

ID	detector	filter	resolver	MWE #	Sibling #	Sibling %
1	Exhaustive	MoreFreqAsMWE	Longest	53,208	42,309	79.51
2	Exhaustive	MoreFreqAsMWE	Leftmost	51,543	21,532	41.85
3	Proper Nouns	no filter	Longest	32,583	28,068	86.14
4	Exhaustive	ConstrainLength	Leftmost	49,587	19,984	40.30
5	Stop words	no filter	Longest	13,623	286	2.09

we can use *model*$_B$ to improve over *model*$_A$ on gold standard$_A$ by using model combination with *model*$_A$ and *model*$_B$. We deal with each of these in turn. We test all evaluation schemes on all of our versions of *model*$_B$. Results fluctuate according to the recognisers as discussed in Section 4.3. We give general remarks about results and report our best results in this Section.

4.2.1 Can representing MWEs as one token introduce a training effect?

In order to find out whether or not training data on an MWE-informed corpus can lead to an improved accuracy, i.e. leads to training effect, we compare the output of *model*$_B$ against the output of *model*$_A$ tested on the "gold test" data. 3 out of our 5 *model*$_B$s outperform *model*$_A$ on unlabeled$_B$, although generally by a slight margin. The best results are obtained by model$_{B_3}$ and are given in Table 2. Model$_B$ significantly outperforms *model*$_A$ by 0.24% ($p = 0.006$) which supports the hypothesis that there is indeed a training effect.

Table 2: Precision (P), recall (R), and F$_1$-measure of unlabelled dependencies against gold standard B with recogniser 3

model	test data	P	R	F$_1$
A	gold test	**84.53**	84.76	84.64
B3	gold test	84.48	**85.28**	**84.88**

Because using gold test data gives *model*$_B$ an unfair advantage, we also test these models on the "fully transformed test" data. In this case 3 of our 5 *model*$_B$s outperform *model*$_A$ again although by an even slighter margin. The biggest difference in results is obtained with model$_{B_1}$ and results are given in Table 3. Although the margin is smaller, *model*$_B$ still significantly outperforms *model*$_A$ by 0.15% ($p = 0.047$) which shows that there is a training effect.

Table 3: Precision (P), recall (R), and F_1-measure of unlabelled dependencies against gold standard B with recogniser 1

model	test data	P	R	F_1
A	fully transformed test	**73.15**	72.38	72.77
B1	fully transformed test	73.08	**72.74**	**72.92**

4.2.2 Can representing MWEs as one token introduce a parsing effect?

In order to test whether there can be a parsing effect, we compare the output of *model*$_A$ when data are transformed before parsing with *model*$_A$ when data are transformed after parsing. In this case *model*$_B$ always outperforms *model*$_A$. Our best results are shown in Table 4 in which *model*$_B$ highly significantly outperforms *model*$_A$ ($p < 0.0001$).

Table 4: Precision (P), recall (R), and F_1-measure of unlabelled dependencies against gold standard B with recogniser 1 when transforming before parsing uses gold sibling information and only siblings are transformed after parsing

model	transformed	P	R	F_1
A	before parsing	**83.88**	**84.24**	**84.06**
A	after parsing	78.92	79.41	79.17

The problem with these results is that "transforming before parsing" method has gold standard information about siblings which the "transforming after parsing" method does not. In order to cross-validate our result, we transform all MWEs both before and after parsing and compare the results. In this case, *model*$_A$ when data are transformed before parsing outperforms *model*$_A$ when data are transformed after parsing only in one of the 5 cases which undermines a little the previous argument about the parsing effects showing that there can also be undesirable effects to transforming test data. It could, however, be partly due to the fact that we transformed non-siblings, which may have triggered errors during parsing. In any case, our best results still show a significant improvement with the "transforming before parsing" method over the "transforming after parsing" method. These results are obtained when using recogniser$_3$ and are given in Table 5. Model$_A$ transformed before parsing significantly outperforms *model*$_A$

transformed after parsing by 0.20% ($p = 0.008$). This indicates that there can be a parsing effect.

Table 5: Precision (P), recall (R), and F_1-measure of unlabelled dependencies against gold standard B with recogniser 3 when all MWEs are considered siblings

model	transformed	P	R	F_1
A	before parsing	**79.83**	79.54	**79.69**
A	after parsing	79.38	**79.60**	79.49

4.2.3 Can we improve the parsing model on the original gold standard?

We now verify if we can also outperform the baseline on the untransformed gold standard. This means testing whether or not *model$_B$* improves over *model$_A$* on *external edges* and/or on *mediating edges*. We test this by combining dependency edges obtained from *model$_A$* and *model$_B$*. We combine these edges with 3 different methods. *Internal* edges are always taken from *model$_A$* and *external edges* are always taken from *model$_B$*. *Mediating edges* are taken from A in the *medFromA* evaluation and from B in the 2 other cases. In the *rightmostMed* evaluation, the rightmost MWE unit is always chosen as incoming or outgoing node and in the *leftmostMed* evaluation, it is the leftmost MWE unit that is always taken as incoming or outgoing node. Our best results are given in Table 6 in which *model$_B$* only outperforms *model$_A$* in the *medFromA* case by 0.13% which is not significant ($p > 0.05$). This seems to show that *model$_B$* may perform better than *model$_A$* on *external edges* but as far as *mediating edges* are concerned, the picture is unclear. If we take the *mediating edge* from B, it seems clearly better to choose the rightmost MWE unit as incoming or outgoing node (which is not surprising since compound nouns are almost always right-headed) but doing so does not seem to be a big help in parsing accuracy. Model$_B$ might perform better than *model$_A$* on *mediating edges* if we had a better mechanism to recover the head word but with our simple method we cannot say whether or not this is the case.

In this result, *model$_B$* is again helped in the parsing decisions by being told which MWEs are siblings. In order to test whether we can improve on *model$_A$* in a fully automatic manner, we test *model$_B$* on the "fully transformed test" data which is a version of the test data obtained automatically, i.e. by transforming all MWEs in the text instead of only the siblings. All MWEs are then parsed as a unit. When we combine the models, we have more MWEs than we should

Table 6: Precision (P), recall (R), and F_1-measure of unlabelled dependencies against gold standard A using recogniser 3

model	combination type	P	R	F_1
A		**85.27**	85.02	85.15
A+B3	medFromA	84.89	**85.68**	**85.28**
A+B3	rightmostMed	84.84	85.46	85.15
A+B3	leftmostMed	81.43	82.02	81.72

have and consequently, more edges are considered to be *mediating* and *internal* edges and less edges are considered to be *external* edges. Hence, we are led to choose edges from $model_A$ where $model_A$ is not expected to perform better than $model_B$. When combining both models with the *medFromA* method, however, we still outperform $model_A$ by 0.04% when using $recogniser_3$ showing that $model_B$ may have learnt something useful although there is no significant evidence for it at this point.

4.3 Does using different MWE recognisers impact parsing accuracy differently?

As explained in Section 3.5.3, the last experiment we conduct is to test our model using different recognisers, combine the output using the model combination algorithm explained in Section 3.5.1 and compare it to gold $standard_A$. This provides a way to compare different versions of our $model_B$.

As can be seen in Table 7, different MWE recognition methods seem to make a difference in results. There is a significant difference between our best model (based on $recogniser_3$) and our worst model (based on $recogniser_2$) of 0.26 (p = 0.01). Some recognisers lead to decreases in parsing accuracy while others lead to increases. It appears from the table that using a leftmost resolver (a resolver that always chooses the leftmost MWE when there is a conflict) has a bad impact on parsing accuracy. Looking at the different models, it is interesting to note that there is a much lower percentage of MWEs that are siblings in the tree and hence a much lower amount of changes made in the treebank. It is interesting to note that the best model is based on a detector that only detects proper nouns. This seems to show that they are the best candidates for being treated as words-with-spaces. This is not surprising because they are not flexible and never get inflected. For other types of MWE, an analysis as word-with-spaces might not be the most appropriate, as argued by many researchers (Sag et al. 2002) to give just one example, see Section 2).

Table 7: F_1-measure of unlabelled dependencies against gold standard A using different recognisers from the *model$_A$* combining method

model	detector type	resolver type	F_1
A			85.15
B1	exhaustive	longest	85.18
B2	exhaustive	leftmost	85.02
B3	Proper Nouns	longest	**85.28**
B4	Length 2	leftmost	85.07
B5	Stop words	longest	85.19

4.4 Summary of our findings

We summarise our findings in Table 8.

Table 8: Summary of our findings

question	answer	tables concerned
Can there be a parsing effect?	yes	Table 2 and 3
Can there be a training effect?	yes	Table 4 and 5
Can we improve parsing on the untransformed gold standard?	not significantly	Table 6
Do different types of MWEs impact the results differently?	yes	Table 7

Table 2 and 3 are respectively upper and lower bounds on the training effect that can be obtained with our method with these recognisers. Similarly, Table 4 and 5 are respectively upper and lower bounds on the parsing effect that can be obtained. Given that the lower bounds are still significantly above the baselines in both cases, we can conclude that there can be both a training and a parsing effect, and that we can improve CCG parsing with information about MWEs.

5 Conclusion

5.1 Contributions

Our main contributions in this work are:

- Improvements on CCG parsing with automatic MWE recognition
- Significant results despite limited settings
- An algorithm to automatically transform MWEs in a treebank
- Techniques for distinguishing training from parsing effects
- Empirical support that there is both training and parsing effects
- Interesting differences in results when using different recognisers

The task we have been trying to improve in this work is the task of syntactic parsing. Adding MWE information to CCG parsing was singled out as a useful direction because it has proven useful in the past with other parsing frameworks and because it seemed an interesting approach to attempt within the framework of a lexicalized grammar. We built on previous work which had shown the benefits of giving information about MWEs to a syntactic parser. It had been shown to work for deterministic dependency parsing, shallow parsing and deterministic constituency parsing but not for statistical constituency parsing. We implemented an existing pipeline which consists in transforming the representation of MWEs in training and test data by collapsing its units to one token and adapted it to our purposes. We gave further evidence supporting these studies and showed that statistical constituency parsing with a lexicalized grammar too can benefit from MWE information. Our study provided further empirical support to the hypothesis that MWE information can improve syntactic parsing by showing that we can improve CCG parsing with information about MWEs.

MWE identification was also identified as a notoriously difficult task although important for many applications because MWEs violate usual compositional rules and can be the source of many errors if not handled properly. We have shown that using an existing automatic recogniser as a source of MWE information was useful which had so far been left a bit unclear in the literature.

Our results have shown small but significant improvements over previous models which is very encouraging given the restricted settings we have worked with. We have as a matter of fact hypothesized that the results were very much limited by the methodology used and have suggested ways of improving the current approach. Our biggest contributions, however, are not in the results we obtained but in the techniques we proposed. The study has proposed novel techniques to improve on previous pipelines. We have proposed an algorithm to automatically transform MWEs in a treebank which can be used with other formalisms although this algorithm is limited to transforming MWEs which form a constituent in the tree. More importantly, we have proposed ways of experimenting with our models in a way that we can distinguish parsing (the parser is helped in its decisions by transformed data) from training effects (the parser learns something useful) in the evaluation and have shown evidence for both. In addition, we have proposed to experiment with different MWE recognisers and study the impact of different MWE recognition methods on parsing accuracy. This is especially interesting in that it is never quite clear in the literature what counts as an MWE. Experimenting with recognisers that detect different types of MWEs can help find out what types of MWEs this method is most suitable for. Our results in this work have shown that collapsing MWE units to one token is most useful for MWEs that are made of proper nouns. It makes intuitive sense that treating them as words with spaces is appropriate since they are not flexible and do not get inflected.

5.2 Future work

We propose the following for future work:

- Extending the transformation algorithm to the non-sibling case
- Testing more MWE recognition methods
- Conducting error analysis

A lot more interesting research can still be done on the interaction between MWE identification and syntactic parsing. Theoretical research has emphasized the need to give different syntactic representations for different types of MWEs but a lot of empirical work is still needed if we want to automatically assign sensible syntactic representations to MWEs. Extending our transformation algorithm to the non-sibling case would allow conducting more extensive experiments. We also believe that testing more recognition methods could lead to interesting discussions where we could find out more about what type of MWE is dealt best with by what method. This could also help discover interesting properties of MWEs. Conducting error analysis could also lead to further insight into why the method is sometimes successful, sometimes less successful.

In the meantime, we believe to have offered new perspectives in the study of the integration between syntactic parsing and MWE identification especially in relation to CCG parsing. We have given encouraging results on a difficult task and suggested ways of improving them. We have given further evidence that the integration of MWE identification with syntactic parsing is a promising and exciting research direction.

References

Birch, Alexandra, Miles Osborne & Philipp Koehn. 2007. CCG supertags in factored statistical machine translation. In *Proceedings of the second workshop on Statistical Machine Translation*, 9–16.

Christodoulopoulos, Christos. 2008. *Creating a natural logic inference system with Combinatory Categorial Grammar*. University of Edinburgh MA thesis.

Clark, Stephen & James Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics* 4(33). 493–552.

Clark, Stephen & Julia Hockenmaier. 2002. Evaluating a wide-coverage CCG parser. In *Proceedings of the LREC 2002 beyond PARSEVAL workshop*, 60–66.

Constable, James & James Curran. 2009. Integrating verb-particle constructions into CCG parsing. In *Proceedings of the Australasian Language Technology Association workshop 2009*, 114–118. Sydney, Australia. http://aclweb.org/anthology/U09-1017.

Constant, Mathieu & Joakim Nivre. 2016. A transition-based system for joint lexical and syntactic analysis. In *Proceedings of the 54th annual meeting of the Association for Computational Linguistics*, vol. 1: *Long papers*, 161–171. Berlin, Germany: Association for Computational Linguistics. http://www.aclweb.org/anthology/P16-1016.

Constant, Mathieu, Anthony Sigogne & Patrick Watrin. 2012. Discriminative strategies to integrate multiword expression recognition and parsing. In *Proceedings of the 50th annual meeting of the Association for Computational Linguistics: Long papers*, vol. 1 (ACL '12), 204–212. Jeju Island, Korea: Association for Computational Linguistics. http://dl.acm.org/citation.cfm?id=2390524.2390554.

Copestake, Ann, Fabre Lambeau, Aline Villavicencio, Francis Bond, Timothy Baldwin, Ivan Sag & Dan Flickinger. 2002. Multiword expressions: Linguistic precision and reusability. In *Proceedings of the third international conference on Language Resources and Evaluation (LREC 2002)*, 1941–1947. Las Palmas, Canary Islands, Spain.

Curran, James, Stephen Clark & Johan Bos. 2007. Linguistically motivated large-scale NLP with C&C and Boxer. In *Proceedings of the 45th annual meeting of the Association for Computational Linguistics, companion volume, proceedings of the demo and poster sessions*, 33–36. Prague, Czech Republic: Association for Computational Linguistics.

Deoskar, Tejaswini, Christos Christodoulopoulos, Alexandra Birch & Mark Steedman. 2014. Generalizing a strongly lexicalized parser using unlabeled data. In Gosse Bouma & Yannick Parmentier (eds.), *Proceedings of the 14th conference of the European chapter of the Association for Computational Linguistics (EACL'2014)*, 126–134. Gothenburg.

Eryiğit, Gülşen, Tugay İlbay & Ozan Arkan Can. 2011. Multiword expressions in statistical dependency parsing. In *Proceedings of the second workshop on Statistical Parsing of Morphologically Rich Languages*, 45–55. Dublin, Ireland.

Finlayson, Mark Alan & Nidhi Kulkarni. 2011. Detecting multi-word expressions improves word sense disambiguation. In *Proceedings of the workshop on Multiword Expressions: From parsing and generation to the real world* (MWE '11), 20–24. Stroudsburg, PA, USA: Association for Computational Linguistics.

Green, Spence, Marie-Catherine de Marneffe & Christopher D. Manning. 2013. Parsing models for identifying multiword expressions. *Computational Linguistics* 39(1). 195–227.

Hockenmaier, Julia. 2003. *Data and models for statistical parsing with Combinatory Categorial Grammar*. University of Edinburgh dissertation.

Hockenmaier, Julia & Mark Steedman. 2002. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th annual meeting on Association for Computational Linguistics* (ACL '02), 335–342. Stroudsburg, PA, USA: Association for Computational Linguistics.

Hockenmaier, Julia & Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics* 33(3). 355–396.

Hoffmann, Thomas & Graeme Trousdale. 2013. Construction Grammar: Introduction. In Thomas Hoffmann & Graeme Trousdale (eds.), *The Oxford handbook of Construction Grammar* (Oxford Handbooks in Linguistics), 1–14. Oxford University Press.

Kim, Su Nam. 2008. *Statistical modeling of multiword expressions*. University of Melbourne dissertation.

Korkontzelos, Ioannis & Suresh Manandhar. 2010. Can recognising multiword expressions improve shallow parsing? In *Proc. of the 11th annual conference of the North American chapter of the Association for Computational Linguistics:*

Human Language Technologies NAACL/HLT 2010, 636–644. Los Angeles, California.

Krishnamurthy, Jayant & Tom M Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 joint conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, 754–765.

Lewis, Mike & Mark Steedman. 2013. Combining distributional and logical semantics. *Transactions of the Association for Computational Linguistics* 1. 179–192.

Martens, Scott & Vincent Vandeghinste. 2010. An efficient, generic approach to extracting multi-word expressions from dependency trees. In *Proceedings of the workshop on Multiword Expressions: From theory to applications (MWE 2010)*, 84–87. Beijing, China: Association for Computational Linguistics.

Nivre, Joakim & Jens Nilsson. 2004. Multiword units in syntactic parsing. In *Workshop on Methodologies and Evaluation of Multiword Units in Real-World Applications*, 39–46.

Nunberg, Geoffrey, Ivan Sag & Thomas Wasow. 1994. Idioms. *Language* 70(3). 491–538.

Padó, Sebastian. 2006. *User's guide to* `sigf`*: Significance testing by approximate randomisation.*

Sag, Ivan, Timothy Baldwin, Francis Bond, Ann Copestake & Dan Flickinger. 2002. Multiword expressions: A pain in the neck for NLP. In *Proceedings of the 3rd international conference on Computational Linguistics and Intelligent Text Processing* (Lecture Notes in Computer Science 2276), 1–15. Springer.

Schneider, Nathan, Spencer Onuffer, Nora Kazour, Emily Danchik, Michael T. Mordowanec, Henrietta Conrad & Noah A. Smith. 2014. Comprehensive annotation of multiword expressions in a social web corpus. In *Proceedings of the ninth international conference on Language Resources and Evaluation (LREC 2014)*, 456–461. Reykyavik.

Seretan, Violeta. 2013. On collocations and their interaction with parsing and translation. *Informatics* 1(1). 11–31.

Steedman, Mark. 2000. *The syntactic process*. Cambridge, MA, USA: MIT Press.

Weller, Marion & Ulrich Heid. 2010. Extraction of German multiword expressions from parsed corpora using context features. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner & Daniel Tapias (eds.), *Proceedings of LREC 2010*, 3195–3201. Valletta: European Language Resources Association.

White, Michael. 2006. Efficient realization of coordinate structures in Combinatory Categorial Grammar. *Research on Language and Computation* 4(1). 39–75.

Zhang, Yi & Valia Kordoni. 2008. Robust parsing with a large HPSG grammar. In *Proceedings of the sixth international Language Resources and Evaluation conference (LREC'08)*, 1888–1893. Marrakech, Morroco.

Zhang, Yi, Valia Kordoni, Aline Villavicencio & Marco Idiart. 2006. Automated multiword expression prediction for grammar engineering. In *Proceedings of the workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties* (MWE '06), 36–44. Stroudsburg, PA, USA: Association for Computational Linguistics.