Deep Lexical Segmentation and Syntactic Parsing in the Easy-First Dependency Framework

Matthieu Constant[♠◇] **Joseph Le Roux**[♣] **Nadi Tomeh**[♣]
[♠] Université Paris-Est, LIGM, Champs-sur-Marne, France
[◇] Alpage, INRIA, Université Paris Diderot, Paris, France
[♣] LIPN, Université Paris Nord, CNRS UMR 7030, Villetaneuse, France
matthieu.constant@u-pem.fr, leroux@lipn.fr, tomeh@lipn.fr

Abstract

We explore the consequences of representing token segmentations as hierarchical structures (trees) for the task of Multiword Expression (MWE) recognition, in isolation or in combination with dependency parsing. We propose a novel representation of token segmentation as trees on tokens, resembling dependency trees. Given this new representation, we present and evaluate two different architectures to combine MWE recognition and dependency parsing in the *easy-first* framework: a pipeline and a joint system, both taking advantage of lexical and syntactic dimensions. We experimentally validate that MWE recognition significantly helps syntactic parsing.

1 Introduction

Lexical segmentation is a crucial task for natural language understanding as it detects semantic units of texts. One of the main difficulties comes from the identification of multiword expressions [MWE] (Sag et al., 2002), which are sequences made of multiple words displaying multidimensional idiomaticity (Nunberg et al., 1994). Such expressions may exhibit syntactic freedom and varying degree of compositionality, and many studies show the advantages of combining MWE identification with syntactic parsing (Savary et al., 2015), for both tasks (Wehrli, 2014). Indeed, MWE detection may help parsing, as it reduces the number of lexical units, and in turn parsing may help detect MWEs with syntactic freedom (syntactic variations, discontinuity, etc.).

In the dependency parsing framework, some previous work incorporated MWE annotations within syntactic trees, in the form of complex subtrees either with flat structures (Nivre and Nilsson, 2004; Eryiğit et al., 2011; Seddah et al., 2013) or deeper ones (Vincze et al., 2013; Candito and Constant, 2014). However, these representations do not capture deep lexical analyses like nested MWEs. In this paper, we propose a two-dimensional representation that separates lexical and syntactic layers with two distinct dependency trees sharing the same nodes[1]. This representation facilitates the annotation of complex lexical phenomena like embedding of MWEs (e.g. *I will (take a (rain check))*). Given this representation, we present two easy-first dependency parsing systems: one based on a pipeline architecture and another as a joint parser.

2 Deep Segmentation and Dependencies

This section describes a lexical representation able to handle nested MWEs, extended from Constant and Le Roux (2015) which was limited to shallow MWEs. Such a lexical analysis is particularly relevant to perform deep semantic analysis.

A lexical unit [LU] is a subtree of the lexical segmentation tree composed of either a single token unit or an MWE. In case of a single token unit, the subtree is limited to a single node. In case of an MWE, the subtree is rooted by its leftmost LU, from which there are arcs to every other LU of the MWE. For instance, the MWE *in spite of* made of three single token units is a subtree rooted by *in*. It comprises two arcs: *in* → *spite* and *in* → *of*. The MWE *make*

[1]This is related to the *Prague Dependency Treebank* (Hajič et al., 2006) which encodes MWEs in tectogrammatical trees connected to syntactic trees (Bejček and Straňák, 2010).

Proceedings of NAACL-HLT 2016, pages 1095–1101,
San Diego, California, June 12-17, 2016. ©2016 Association for Computational Linguistics

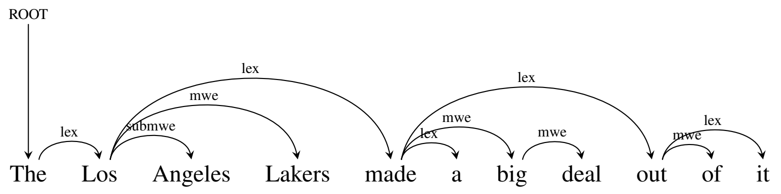

Figure 1: Deep segmentation of *Los Angeles Lakers made a big deal out of it* represented as a tree.

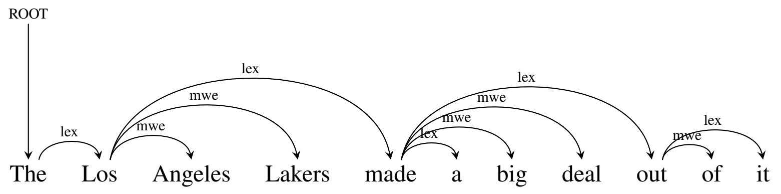

Figure 2: Shallow segmentation of *Los Angeles Lakers made a big deal out of it* represented as a tree.

big deal is more complex as it is formed of a single token unit *make* and an MWE *big deal*. It is represented as a subtree whose root is *make* connected to the root of the MWE subtree corresponding to *big deal*. The subtree associated with *big deal* is made of two single token units. It is rooted by *big* with an arc *big* → *deal*. Such structuring allows to find nested MWEs when the root is not an MWE itself, like for *make big deal*. It is different for the MWE *Los Angeles Lakers* comprising the MWE *Los Angeles* and the single token unit *Lakers*. In that case, the subtree has a flat structure, with two arcs from the node *Los*, structurally equivalent to *in spite of* that has no nested MWEs. Therefore, some extra information is needed in order to distinguish these two cases. We use arc labels.

Labeling requires to maintain a counter l in order to indicate the embedding level in the leftmost LU of the encompassing MWE. Labels have the form $sub^l mwe$ for $l \geq 0$. Let $U = U_1...U_n$ be a LU composed of n LUs. If $n = 1$, it is a single token unit. Otherwise, $subtree(U, 0)$, the lexical subtree[2] for U is recursively constructed by adding arcs $subtree(U_1, l+1) \xrightarrow{sub^l mwe} subtree(U_i, 0)$ for $i \neq 1$. In the case of *shallow* representation, every LUs of U are single token units.

Once built the LU subtrees (the *internal dependencies*), it is necessary to create arcs to connect them and form a complete tree : that we call *external dependencies*. LUs are sequentially linked together: each pair of consecutive LUs with roots (w_i,w_j), $i < j$, gives an arc $w_i \xrightarrow{lex} w_j$. Figure 1 and Figure 2 respectively display the deep and shallow lexical segmentations of the sentence *The Los Angeles Lakers made a big deal out of it*.

For readibility, we note mwe for $sub^0 mwe$ and $submwe$ for $sub^1 mwe$.

3 Multidimensional Easy-first Parsing

3.1 Easy-first parsing

Informally, easy-first proposed in Goldberg and Elhadad (2010) predicts easier dependencies before risky ones. It decides for each token whether it must be attached to the root of an adjacent subtree and how this attachment should be labeled[3]. The order in which these decisions are made is not decided in advance: highest-scoring decisions are made first and constrain the following decisions.

This framework looks appealing in order to test our assumption that segmentation and parsing are mutually informative, while leaving the exact flow of information to be learned by the system itself: we do not postulate any priority between the tasks nor that all attachment decisions must be taken jointly. On the contrary, we expect most decisions to be made independently except for some difficult cases that need both lexical and syntactic knowledge.

We now present two adaptations of this strategy to

[2]The second argument l corresponds to the embedding level.

[3]Labels are an extension to Goldberg and Elhadad (2010)

build both lexical and parse trees from a unique sequence of tokens[4]. The key component is to use features linking information from the two dimensions.

3.2 Pipeline Architecture

In this trivial adaptation, two parsers are run sequentially. The first one builds a structure in one dimension (i.e. for segmentation or syntax). The second one builds a structure in the other dimension, with the result of the first parser available as features.

3.3 Joint Architecture

The second adaptation is more substantial and takes the form of a joint parsing algorithm. This adaptation is provided in Algorithm 1. It uses a single classifier to predict lexical and syntactic actions. As in easy-first, each iteration predicts the most certain head attachment action given the currently predicted subtrees, but here it may belong to any dimension. This action can be mapped to an edge in the appropriate dimension via function EDGE. Function *score(a,i)* computes the dot-product of feature weights and features at position i using surrounding subtrees in both dimensions[5].

Algorithm 1 Joint Easy-first parsing

```
 1: function JOINT EASY-FIRST PARSING($w_0 ... w_n$)
 2:     Let $\mathcal{A}$ be the set of possible actions
 3:     arcs$_s$,arcs$_l$ := $(\emptyset, \emptyset)$
 4:     h$_s$,h$_l$ := $w_0 \dots w_n, w_0 \dots w_n$
 5:     while $|h_l| > 1 \vee |h_s| > 1$ do
 6:         $\hat{a}, \hat{i}$ := argmax$_{a \in \mathcal{A}, i \in [|h_d|]}$ score(a,i)
 7:         $(par, lab, child, dim)$ := EDGE$((h_s, h_l), \hat{a}, \hat{i})$
 8:         arcs$_{dim}$ := arcs$_{dim} \cup (par, lab, child)$
 9:         h$_{dim}$ := h$_{dim} \backslash \{child\}$
10:     end while
11:     return $(arcs_l, arcs_s)$
12: end function
13: function EDGE$((h_s, h_l), (dir, lab, dim), i)$
14:     if $dir = \leftarrow$ then                ▷ we have a left edge
15:         return $(h_{dim}[i], lab, h_{dim}[i+1], dim)$
16:     else
17:         return $(h_{dim}[i+1], lab, h_{dim}[i], dim)$
18:     end if
19: end function
```

We can reuse the reasoning from Goldberg and Elhadad (2010) and derive a worst-case time complexity of $O(n \log n)$, provided that we restrict feature extraction at each position to a bounded vicinity.

[4]It is straightforward to add any number of tree structures.

[5]Let us note that the algorithm builds projective trees for each dimension, but their union may contain crossing arcs.

	English	French	
Corpus	**EWT**	**FTB**	**Sequoia**
# words	55,590	564,798	33,829
# MWE labels	4,649	49,350	6,842
ratio	0.08	0.09	0.20
MWE rep.	shallow$^+$	shallow	deep

Table 1: Datasets statistics. The first part describes the number of words in training sets with MWE label ratio. shallow$^+$ refers to a shallow representation with enriched MWE labels indicating the MWE strength (collocation vs. fixed).

4 Experiments

4.1 Datasets

We used data sets derived from three different reference treebanks: English Web Treebank (Linguistic Data Consortium release LDC2012T13)[EWT], French treebank (Abeillé et al., 2003) [FTB], Sequoia Treebank (Candito and Seddah, 2012) [Sequoia]. These treebanks have MWE annotations available on at least a subpart of them. For EWT, we used the STREUSLE corpus (Schneider et al., 2014b) that contains annotations of all types of MWEs, including discontiguous ones. We used the train/test split from Schneider et al. (2014a). The FTB contains annotations of contiguous MWEs. We generated the dataset from the version described in Candito and Constant (2014) and used the shallow lexical representation, in the official train/dev/test split of the SPMRL shared task (Seddah et al., 2013). The Sequoia treebank contains some limited annotations of MWEs (usually, compounds having an irregular syntax). We manually extended the coverage to all types of MWEs including discontiguous ones. We also included deep annotation of MWEs (in particular, nested ones). We used a 90%/10% train/test split in our experiments. Some statistics about the data sets are provided in table 4.1. Tokens were enriched with their predicted part-of-speech (POS) and information from MWE lexicon[6] lookup as in Candito and Constant (2014).

[6]We used the *Unitex* platform (`www-igm.univ-mlv.fr/~unitex/` for French and the STREUSLE corpus web site (`www.ark.cs.cmu.edu/LexSem/`) for English.

4.2 Parser and features

Parser. We implemented our systems by modifying the parser of Y. Goldberg[7] also used as a baseline. We trained all models for 20 iterations with dynamic oracle (Goldberg and Nivre, 2013) using the following exploration policy: always choose an oracle transition in the first 2 iterations ($k = 2$), then choose model prediction with probability $p = 0.9$.

Features. One-dimensional features were taken directly from the code supporting Goldberg and Nivre (2013). We added information on typographical cues (hyphenation, digits, capitalization, ...) and the existence of substrings in MWE dictionaries in order to help lexical analysis. Following Constant et al. (2012) and Schneider et al. (2014a), we used dictionary lookups to build a first naive segmentation and incorporate it as a set of features. Two-dimensional features were used in both pipeline and joint strategies. We first added syntactic path features to the lexical dimension, so syntax can guide segmentation. Conversely, we also added lexical path features to the syntactic dimension to provide information about lexical connectivity. For instance, two nodes being checked for attachment in the syntactic dimension can be associated with information describing whether one of the corresponding node is an ancestor of the other one in the lexical dimension (i.e. indicating whether the two syntactic nodes are linked via internal or external paths).

We also selected automatically generated features combining information from both dimensions. We chose a simple data-driven heuristics to select combined features. We ran one learning iteration over the FTB training corpus adding all possible combinations of syntactic and lexical features. We picked the templates of the 10 combined features whose scores had the greatest absolute values. Although this heuristics may not favor the most discriminant features, we found that the chosen features helped accuracy on the development set.

4.3 Results

For each dataset, we carried out four experiments. First we learned and ran independently two *distinct* baseline easy-first parsers using one-dimensional features: one producing a lexical segmentation, another one predicting a syntactic parse tree. We also trained and ran a *joint* easy-first system predicting lexical segmentations and syntactic parse trees, using two-dimensional features. We also experimented the *pipeline* system for each dimension, consisting in applying the baseline parser on one dimension and using the resulting tree as source of two-dimensional features in a standard easy first parser applied on the other dimension. Since pipeline architectures are known to be prone to error propagation, we also run an experiment where the pipeline second stage is fed with oracle first-stage trees.

Results on the test sets are provided in table 2, where LAS and UAS are computed with punctuation. Overall, we can see that the lexical information tends to help syntactic prediction while the other way around is unclear.

	Syntactic		**Lexical**		
Model	**UAS**	**LAS**	**UAS**	**LAS**	**F1 (Pr / Rc)**
			FTB		
Distinct	87.44	85.09	96.69	94.75	79.47 (81.18/77.83)
Pipeline	**87.74**	**85.39**†	96.74	94.83	79.82 (81.56/78.15)
–oracle trees	88.96	86.98†	97.89	96.62†	87.27 (87.78/86.76)
Joint	87.69	85.32†	**96.79**	**94.89**	**80.11** (82.51/77.85)
Le Roux et al. (2014) CRF					80.49
Le Roux et al. (2014) combination					82.44
Candito and Constant (2014) graph-based parsing + CRF					
	89.24	86.97			78.60
			Sequoia		
Distinct	84.88	81.74	**89.70**	**85.00**	67.60 (73.56/62.53)
Pipeline	85.91	82.84†	89.57	84.70	67.04 (72.24/62.53)
–oracle trees	85.95	83.05†	90.03	85.64†	69.36 (75.23/64.34)
Joint	**86.19**	**82.99**†	89.32	84.76	**68.58** (72.75/64.86)
			EWT		
Distinct	87.45	83.91	93.96	90.75	**53.93** (66.42/45.39)
Pipeline	**88.45**	**84.76**†	**94.02**	**90.80**	53.19 (68.09/43.64)
–oracle trees	88.20	84.76†	94.23	91.09	55.05 (71.15/44.89)
Joint	87.98	84.24	93.72	90.49	51.20 (64.64/42.39)
Schneider et al. (2014a) Baseline					53.85 (60.99/48.27)
Schneider et al. (2014a) Best (oracle POS and clusters)					57.71 (58.51/57.00)

Table 2: Results on our three test sets. Statistically significant differences (p-value < 0.05) from the corresponding "distinct" setting are indicated with †. Rows *-oracle trees* are the same as pipeline but using oracle, instead of predicted, trees.

5 Discussion

The first striking observation is that the syntactic dimension does not help the predictions in the lexical dimension, contrary to what could be expected. In practice, we can observe that variations and discontinuity of MWEs are not frequent in our data sets. For instance, Schneider et al. (2014a) notice

[7]We started from the version available at the time of writing at `https://bitbucket.org/yoavgo/tacl2013dynamicoracles`

that only 15% of the MWEs in EWT are discontiguous and most of them have gaps of one token. This could explain why syntactic information is not useful for segmentation. On the other hand, the lexical dimension tends to help syntactic predictions. More precisely, while the pipeline and the joint approach reach comparable scores on the FTB and Sequoia, the joint system has disappointing results on EWT. The good scores for Sequoia could be explained by the larger MWE coverage.

In order to get a better intuition on the real impact of each of the three approaches, we broke down the syntax results by dependency labels. Some labels are particularly informative. First of all, the precision on the modifier label *mod*, which is the most frequent one, is greatly improved using the pipeline approach as compared with the baseline (around 1 point). This can be explained by the fact that many nominal MWEs have the form of a regular noun phrase, to which its internal adjectival or prepositional constituents are attached with the *mod* label. Recognizing a nominal MWE on the lexical dimension may therefore give a relevant clue on its corresponding syntactic structure. Then, the *dep_cpd* connects components of MWE with irregular syntax that cannot receive standard labels. We can observe that the pipeline (resp. the joint) approach clearly improves the precision (resp. recall) as compared with the baseline (+1.6 point). This means that the combination of a preliminary lexical segmentation and a possibly partial syntactic context helps improving the recognition of syntax-irregular MWEs. Coordination labels (*dep.coord* and *coord*) are particularly interesting as the joint system outperforms the other two on them. Coordination is known to be a very complex phenomenon: these scores would tend to show that the lexical and syntactic dimensions mutually help each other.

When comparing this work to state-of-the-art systems on data sets with shallow annotation of MWEs, we can see that we obtain MWE recognition scores comparable to systems of equivalent complexity and/or available information. This means that our novel representation which allows for the annotation of more complex lexical phenomena does not deteriorate scores for shallow annotations.

	gold	**distinct**		**pipeline**		**joint**	
Label	**count**	**recall**	**prec.**	**recall**	**prec.**	**recall**	**prec.**
mod	**7782**	**80.39**	**78.18**	**80.62**	**79.13**	**80.94**	**78.68**
obj.p	6247	96.86	96.43	96.70	96.56	96.69	96.44
det	5269	97.67	97.89	97.70	97.76	97.76	97.72
ponct	4682	71.94	71.98	72.32	72.57	72.53	72.35
dep	3350	84.66	83.98	84.72	83.35	84.90	83.67
suj	2044	90.66	92.93	91.39	92.70	91.39	93.49
obj	1716	88.29	87.98	88.69	87.52	88.11	88.52
dep_cpd	**1604**	**84.66**	**87.84**	**86.28**	**87.54**	**85.10**	**89.39**
root	1235	92.23	92.23	92.79	92.79	92.96	92.96
dep.coord	**931**	**83.89**	**83.80**	**83.46**	**84.73**	**83.46**	**85.48**
coord	**832**	**58.77**	**59.27**	**60.10**	**60.39**	**59.98**	**60.71**
aux.tps	516	97.09	99.40	97.67	99.41	97.29	99.41
a_obj	398	75.13	77.06	73.37	79.56	73.62	78.98
obj.cpl	367	83.11	83.79	84.20	84.20	84.74	83.83
ats	345	79.71	83.33	79.42	82.78	79.42	83.03
mod.rel	334	70.96	76.21	70.36	73.90	68.26	73.55
de_obj	329	75.08	74.62	76.60	77.30	75.38	76.07
p_obj	268	58.58	79.70	61.19	79.61	60.45	80.60
aff	245	84.90	79.09	86.53	79.70	88.57	78.06
aux.pass	242	95.04	95.44	94.63	95.02	94.21	95.00
ato	30	33.33	83.33	40.00	85.71	43.33	86.67
arg	22	50.00	68.75	59.09	65.00	59.09	59.09
aux.caus	21	85.71	94.74	85.71	94.74	85.71	94.74
comp	11	0.00	0.00	0.00	0.00	0.00	0.00

Table 3: Results on FTB development set, broken down by dependency labels. Scores correspond to recall and precision.

6 Conclusions and Future Work

In this paper we presented a novel representation of deep lexical segmentation in the form of trees, forming a dimension distinct from syntax. We experimented strategies to predict both dimensions in the easy-first dependency parsing framework. We showed empirically that joint and pipeline processing are beneficial for syntactic parsing while hardly impacting deep lexical segmentation.

The presented combination of parsing and segmenting does not enforce any structural constraint over the two trees[8]. We plan to address this issue in future work. We will explore less redundant, more compact representations of the two dimensions since some annotations can be factorized between the two dimensions (e.g. MWEs with irregular syntax) and some can easily be induced from others (e.g. sequential linking between lexical units).

Acknowledgments

This work has been partly funded by the French *Agence Nationale pour la Recherche*, through the PARSEME-FR project (ANR-14-CERA-0001) and as part of the *Investissements d'Avenir* program (ANR-10-LABX-0083)

[8]for instance, aligned arc or subtrees

References

Anne Abeillé, Lionel Clément, and François Toussenel. 2003. Building a treebank for French. In Anne Abeillé, editor, *Treebanks*. Kluwer, Dordrecht.

Eduard Bejček and Pavel Straňák. 2010. Annotation of multiword expressions in the prague dependency treebank. *Language Resources and Evaluation*, 44(1-2).

Marie Candito and Matthieu Constant. 2014. Strategies for contiguous multiword expression analysis and dependency parsing. In *ACL 14-The 52nd Annual Meeting of the Association for Computational Linguistics*. ACL.

Marie Candito and Djamé Seddah. 2012. Le corpus Sequoia : annotation syntaxique et exploitation pour l'adaptation d'analyseur par pont lexical. In *TALN 2012 - 19e conférence sur le Traitement Automatique des Langues Naturelles*, Grenoble, France.

Matthieu Constant and Joseph Le Roux. 2015. Dependency representations for lexical segmentation. In *Proceedings of the international workshop on statistical parsing of morphologically-rich languages (SPMRL 2015)*.

Matthieu Constant, Anthony Sigogne, and 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 (ACL'12)*, pages 204–212.

Gülşen Eryiğit, Tugay İlbay, and Ozan Arkan Can. 2011. Multiword Expressions in Statistical Dependency Parsing. In *Proceedings of the Second Workshop on Statistical Parsing of Morphologically Rich Languages*, SPMRL '11, pages 45–55, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 742–750. Association for Computational Linguistics.

Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *Transactions of the association for Computational Linguistics*, 1:403–414.

J. Hajič, J. Panevová, E. Hajičová, P. Sgall, P. Pajas, Štěpánek, Havelka J., Mikulová J., Z. M., Žabokrtský, and M. Ševčíková Razímová. 2006. Prague dependency treebank 2.0. *Linguistic Data Consortium*.

Joseph Le Roux, Antoine Rozenknop, and Matthieu Constant. 2014. Syntactic parsing and compound recognition via dual decomposition: Application to french. In *COLING*.

Joakim Nivre and Jens Nilsson. 2004. Multiword units in syntactic parsing. *Proceedings of Methodologies and Evaluation of Multiword Units in Real-World Applications (MEMURA)*.

Geoffrey Nunberg, Ivan A. Sag, and Thomas Wasow. 1994. Idioms. *Language*, 70:491 – 538.

Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for nlp. In *In Proc. of the 3rd International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2002*, pages 1–15.

Agata Savary, Manfred Sailer, Yannick Parmentier, Michael Rosner, Victoria Rosén, Adam Przepiórkowski, Cvetana Krstev, Veronika Vincze, Beata Wójtowicz, Gyri Smørdal Losnegaard, Carla Parra Escartín, Jakub Waszczuk, Matthieu Constant, Petya Osenova, and Federico Sangati. 2015. PARSEME – PARSing and Multiword Expressions within a European multilingual network. In *7th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics (LTC 2015)*, Poznań, Poland, November.

Nathan Schneider, Emily Danchik, Chris Dyer, and Noah A Smith. 2014a. Discriminative lexical semantic segmentation with gaps: running the mwe gamut. *Transactions of the Association for Computational Linguistics*, 2:193–206.

Nathan Schneider, Spencer Onuffer, Nora Kazour, Emily Danchik, Michael T. Mordowanec, Henrietta Conrad, and Noah A. Smith. 2014b. Comprehensive annotation of multiword expressions in a social web corpus. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pages 455–461, Reykjavík, Iceland, May. ELRA.

Djamé Seddah, Reut Tsarfaty, Sandra Ḱubler, Marie Candito, Jinho Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiorkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clérgerie. 2013. Overview of the spmrl 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the 4th Workshop on Statistical Parsing of Morphologically Rich Languages*, Seattle, WA.

Veronika Vincze, János Zsibrita, and Istvàn Nagy T. 2013. Dependency parsing for identifying hungarian light verb constructions. In *Proceedings of International Joint Conference on Natural Language Processing (IJCNLP 2013)*, Nagoya, Japan.

Eric Wehrli. 2014. The relevance of collocations for parsing. In *Proceedings of the 10th Workshop on Multiword Expressions (MWE)*, pages 26–32, Gothenburg, Sweden, April. Association for Computational Linguistics.