Disambiguating Verbs by Collocation: Corpus Lexicography meets Natural Language Processing

Ismail El Maarouf, Vít Baisa, Jane Bradbury, Patrick Hanks
University of Wolverhampton, Masaryk University, University of Wolverhampton, University of Wolverhampton
i.el-maarouf@wlv.ac.uk, xbaisa@fi.muni.cz, J.Bradbury3@wlv.ac.uk, patrick.w.hanks@gmail.com

Abstract

This paper reports the results of Natural Language Processing (NLP) experiments in semantic parsing, based on a new semantic resource, the Pattern Dictionary of English Verbs (PDEV) (Hanks, 2013). This work is set in the DVC (Disambiguating Verbs by Collocation) project aimed at expanding PDEV to a large scale. This project springs from a long-term collaboration of lexicographers with computer scientists which has given rise to the design and maintenance of specific, adapted, and user-friendly editing and exploration tools. Particular attention is drawn on the use of NLP deep semantic methods to help in data processing. Possible contributions for NLP include pattern disambiguation, the focus of this article. The present article explains how PDEV differs from other lexical resources and describes its structure in detail. It also presents new classification experiments on a subset of 25 verbs. The SVM model obtained a micro-average F1 score of 0.81.

Keywords: Corpus Pattern Analysis, Word Sense Disambiguation, Lexical Semantics

1. Introduction

This paper reports the results of Natural Language Processing (NLP) experiments in semantic parsing, based on a new semantic resource, the Pattern Dictionary of English Verbs (PDEV) (Hanks, 2013). This work is set in the DVC (Disambiguating Verbs by Collocation) project[1], a project in corpus lexicography aimed at expanding PDEV to a large scale. This project springs from a long-term collaboration of lexicographers with computer scientists which has given rise to the design and maintenance of specific, adapted, and user-friendly editing and exploration tools. Particular attention is drawn on the use of NLP deep semantic methods to help in data processing. Possible contributions for NLP include pattern disambiguation, the focus of this article.
The present article explains how PDEV differs from other lexical resources and describes its structure in detail (section 2). Section 3 is concerned with pattern disambiguation. Contributions and perspectives are discussed in section 4.

2. Background

2.1. Corpus Pattern Analysis

PDEV is the output of Corpus Pattern Analysis (CPA; (Hanks, 2004)), a technique in corpus lexicography for mapping meaning onto words in text. CPA analyses the prototypical syntagmatic patterns with which words in use are associated. Patterns emerge from the analysis of corpus concordance lines in search for contextual clues which characterize elements of a pattern and which distinguish a pattern from another. Only in a second step is a "meaning" mapped onto a pattern: CPA is driven by syntagmatic patterns, not meaning.
Lexical resources such as PDEV face several issues, mainly *corpus evidence*, *sense granularity*, and *entry coverage*.

- One challenge for lexical resources is to report senses which are actually used in corpora: PDEV is synchronized with a subpart (spoken language documents have been excluded) of the British National Corpus[2] (BNC).
- Sense divisions are difficult to make and to agree on: PDEV lists patterns observable in corpora, and to which only one meaning can be associated.
- Large coverage of entries is hard to achieve: PDEV only contains about 1000 verb entries. The aim of the DVC project is to scale that figure up to 3000, about half of the estimated number of English verbs.

2.2. PDEV

PDEV is maintained with three main tools: the Sketch Engine[3] (Kilgarriff et al., 2004), the CPA editor and the PDEV database[4]. The corpus used is the BNC, a large reference corpus containing various text types in British English (100 million words).
Lexicographers extract typical phraseological patterns from corpora by clustering corpus tokens (tagging them) according to the similarity of their context. The similarity is evaluated in different steps, and consists mostly of syntactic and semantic analysis.

- Syntactic analysis involves the identification of the main structures such as idiomatic expressions, phrasal uses, transitive/intransitive patterns, causative/inchoative alternations, and argument/adjunct discrimination.
- These structures are then iteratively refined, split or joined with others through the analysis of the semantic features shared by collocates in each argument position. For example, Semantic Types (ST; e.g. [[Human]], [[Building]], [[Event]]) are used to represent the prototypical properties shared by the collocates

[1] http://clg.wlv.ac.uk/projects/DVC/

[2] http://www.natcorp.ox.ac.uk/

[3] https://www.sketchengine.co.uk

[4] http://deb.fi.muni.cz/pdev

nb	%	Pattern & *primary implicature*
1	7.14%	[[Plant]] blossom [NO OBJ] *[[Plant]] produces flowers or masses of flowers*
2	92.86%	[[Eventuality | Psych | Human]] blossom [NO OBJ] (into [[Anything = Good]]) *[[Eventuality | Psych | Human]] develops in a promising way or into something that is expected or desired*

Table 1: Patterns for the verb *blossom*

found in a specific pattern position. Semantic categorization is not an obvious task: it requires to identify semantic values shared by arguments, as well as cases of regular polysemy (Pustejovsky, 1995), and decide for a relevant level of generalization.

Patterns can be described according to five types of arguments: Subject, Object, Complement, Adverbial, and Indirect Object. Each can be further detailed using determiners, semantic types, contextual roles, and lexical sets:

- Determiners are used to account for distinctions between "*take place*" and "*take **his** place*".
- Semantic types account for distinctions such as "building [[**Machines**]]" and "building [[**Relationship**]]".
- Contextual roles account for distinctions such as "[[Human=**Film Director**]] shoot" and "[[Human=**Sports Player**]] shoot".
- Lexical sets account for distinctions such as "reap the **whirlwind**" and "reap the **harvest**".

Table 1 shows an example of the PDEV entry of the verb *to blossom*. The first pattern (example 1) is characterized as an intransitive use, having the semantic type [[Plant]] as subject. The main feature which discriminates it from the second pattern is the ST of the subject ([[Eventuality]], [[Psych]], or [[Human]]). Pattern 2 also includes an optional prepositional phrase as an additional context clue.

(1) The Times noted **fruit trees** which had begun to *blossom*...
(2) The **silk trade** *blossomed* in Blockley...

Pattern 2 (example 2) illustrates an alternation of semantic types. It means that in the whole set of lines tagged as "blossom 2", subjects are either [[Eventuality]], [[Psych]] or [[Human]]. A ST provides the relevant semantic value of all words in context. They are, in practice, controlled generalizations of corpus observations. Semantic alternations are used to limit the over-generalization which would result from selecting the highest common node in the ontology.
Each pattern is additionally described with (i) a primary implicature which elaborates the meaning of the pattern, relevant information such as the register, or the domain of use of the pattern, and (ii) percentages. Percentages are obtained by dividing the number of lines tagged with the pattern number over the size of the sample (usually 250 lines).

3. Pattern disambiguation

PDEV has not yet been used in NLP applications, but it could contribute to a large number of them. Preliminary experiments have been made on previous releases of the resource to learn PDEV pattern taggers automatically (see Holub et al., Popescu, El Maarouf et al. 2013).

3.1. Word Sense Disambiguation

Pattern disambiguation is related to Word Sense Disambiguation (WSD). As each pattern is linked to a set of corpus lines, PDEV can be used as a WSD resource, like the SemCor[5] corpus based on WordNet[6]. WSD tasks (Navigli, 2009) require systems to identify the senses of a *word* in a corpus, on the basis of a list of senses (e.g. WordNet synsets) used in a sample of tagged data. The task, in the case of PDEV, would rather be called *Pattern Disambiguation* (PD), in which systems should discriminate pattern numbers, rather than sense IDs. However, the systems, features, and models may be completely different from those used in WSD.
PDEV does not yet compare to the coverage of entries in Wordnet (if WordNet contains 150,000, it is 150 times bigger), but the PDEV corpus compares to SemCor, the largest corpus tagged with WordNet synsets, in terms of the number of tagged tokens: PDEV contains more than 120,000 and SemCor contains the double, about 230,000. The SemCor corpus contrasts with the PDEV tagged corpus in that it is fully tagged (content words) whereas the latter is a targeted lexical sample. Both corpora compare in terms of text type variety and date, but differ in terms of variety of English and total size.
PD systems could be particularly relevant to Statistical Machine Translation, where there is a growing need in multilingual, but also monolingual, semantic tools and resources. Work on phrase-based SMT, such as (Carpuat and Wu, 2007), suggest that work on PD, rather than word disambiguation, may improve SMT systems performance. The following subsections describe experiments led in PD.

3.2. Experiment setting

Pattern disambiguation involves identifying appropriate features from the corpus or external resources, and training a system to predict each verb-specific pattern.
To enable in-depth error analysis, the experiment described here was carried out on a small set of verbs (25 verbs), with high frequency in the BNC, and with a range of different pattern numbers. The created dataset contains 20,336 verb tokens in use, and was split with stratified sampling.

[5] http://www.cse.unt.edu/~rada/downloads.html#semcor
[6] https://wordnet.princeton.edu

The main evaluation metric used was F-score (F1, eq. 4). As pattern disambiguation is a multi-label classification task, it is equally important to know how a system performs across the dataset, as it is to know how well it discriminates each class. A lot of verbs in the dataset have a highly skewed distribution: the baseline applies the most frequent class found in the training set to the test set and often gets a high F1, although it does not perform any pattern discrimination. To reflect this dual aspect, F1 can be averaged in two ways:

- Micro-average gives equal weight to each verb token i of the test set T, and favours systems with the highest number of correct returns (eq. 1–3).
- Macro-average gives equal weight to each verb class k of the class set C, and penalizes systems which only return results for a small number of classes (eq. 5).

$$\text{Precision}_{mic} = \frac{\sum_{i=1}^{|T|} TP_i}{\sum_{i=1}^{|T|}(TP_i + FP_i)} \quad (1)$$

with TP = true positives, FP = false positives

$$\text{Recall}_{mic} = \frac{\sum_{i=1}^{|T|} TP_i}{\sum_{i=1}^{|T|}(TP_i + FN_i)} \quad (2)$$

with TP = true positives, FN = false negatives

$$\text{Micro-F1} = \frac{2 \times Precision_{mic} \times Recall_{mic}}{Precision_{mic} + Recall_{mic}} \quad (3)$$

$$\text{F1} = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (4)$$

$$\text{Macro-F1} = \frac{1}{|C|} \sum_{k=1}^{|C|} .\text{F1}_k \quad (5)$$

Two systems have been tested on this experiment: a Support Vector Machine-based classifier and a boostrapping algorithm based on distributional similarity.

3.3. The Support Vector Machine model

Support Vector Machines (SVM) have been chosen for this task because they generally achieve the best results in WSD (Lee and Ng, 2002). The implementation is based on the Weka[7] SMO module, which approaches polynominal classification by a 1 VS 1 method (pairwise).

Parameter optimisation has been performed through a standard grid search using a linear kernel (more complex methods such as polynomial kernel and radial base functions did not show any improvements in preliminary experiments). The hyperparameter C has been tested with the values {0.1,1,10,100} on each verb training dataset (5-fold cross-validation) and the most accurate models have been selected for testing.

The corpus was parsed with the Stanford CoreNLP module[8], in order to retrieve syntactic dependencies (De Marneffe et al., 2006), POS (Toutanova and Manning, 2000) and named entity tags (Finkel et al., 2005). Lemma, Parts Of Speech, Named Entities and dependency relation names (collapsed), where relevant, have been extracted for four classes of dependency positions: target token, immediate governor, immediate dependent, and dependent of the dependent. Table 2 lists the features extracted from example 2 (illustrated in Fig. 1). Only those features which have been identified in the training set of more than 2 verbs are kept, that is, 4934 features in total (some features are illustrated in Table 3). This criterion was set to extract features which generalise across verbs.

[7] http://www.cs.waikato.ac.nz/ml/weka/

[8] http://nlp.stanford.edu/software/corenlp.shtml

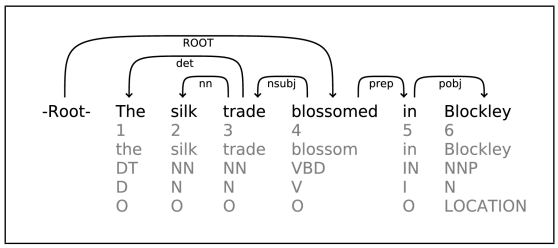

Figure 1: Dependency graph of example 2

Feature id	Dependency structure	Category	Example
1	KWIC	NE	KNE|O
2	KWIC	Lemma	KL|blossom
3	KWIC	POS	KP|VBD
4	KWIC	Relation	None
5	Governor	NE	None
6	Governor	Lemma	None
7	Governor	POS	None
8	Governor	Relation	GR|ROOT
9	Dependent	NE	DNE|nsubj|O
10	Dependent	Lemma	DL|nsubj|trade
11	Dependent	POS	DP|nsubj|NN
12	Dependent	Relation	DR|nsubj
13	Dependent	NE	DNE|prep-in|LOCATION
14	Dependent	Lemma	DL|prep-in|Blockley
15	Dependent	POS	DP|prep-in|NNP
16	Dependent	Relation	DR|prep-in
17	Sub-dependent	NE	DDNE|det|O
18	Sub-dependent	Lemma	DDL|det|the
19	Sub-dependent	POS	DDP|det|DT
20	Sub-dependent	Relation	DDR|det
21	Sub-dependent	NE	DDNE|nn|O
22	Sub-dependent	Lemma	DDL|nn|silk
23	Sub-dependent	POS	DDP|nn|NN
24	Sub-dependent	Relation	DDR|nn
...			

Table 2: Features extracted from example 2

Feature	nb of verbs	gloss
DDP|dobj|PRP	25	*pronoun as a direct object of a dependent*
DP|conj|VB	25	*verb linked with a conjunction*
DDNE|mark|O	25	*null Named Entity as a subordinating conjunction*
KR|dep	25	*dep as a relation of the KWIC*
KR|rcmod	25	*rcmod as a relation of the KWIC*
KR|pcomp	25	*pcomp as a relationof the KWIC*

Table 3: Examples of extracted features

3.4. The bootstrapping model

The second system is a solution available in the Sketch Engine (Kilgarriff and Rychly, 2010). This method boostraps from an existing automatic thesaurus (Grefenstette, 1994; Lin, 1998) to assign a label to a given verb token. The thesaurus is based on a regular grammar which identifies collocates linked to a verb through a syntactic relation (such as *subject*). More formally, dependency triples of the form $||w, r, w'||$, where w is a lemma linked to another lemma w' by a relation r, are filtered by the grammar. Each triple

$$AScore(w, r, w') = \log \frac{||w, r, w'|| \cdot ||*, *, *||}{||w, r, *|| \cdot ||*, *, w'||} \cdot \log(||w, r, w'|| + 1) \quad (6)$$

$$Dist(w, w') = \frac{\sum_{(tuple_i, tuple_j) \in \{tuple_w \cap tuple_{w'}\}} AS_i + AS_j - (AS_i - AS_j)^2/50}{\sum_{tuple_i \in \{tuple_i \cup tuplej\}} AS_i} \quad (7)$$

$$\text{score}_{v,k} = \sum_{(w,r)} \sum_{(w',r)} \max \left(\text{Dist}_{w,w'} \cdot \frac{\sum_{(w,r,k)}}{\sum_{(w,r)}} \right) \quad (8)$$

verb	pattern	train	test	Macro-average			Micro-average		
				A	B	C	A	B	C
arm	4	493	122	0.14	**0.75**	0.62	0.57	**0.74**	0.66
arouse	3	623	156	0.32	**0.57**	0.39	**0.95**	0.89	0.94
arrest	3	1647	412	0.32	0.73	**0.83**	0.97	0.90	**0.98**
beg	13	423	108	0.02	0.52	**0.58**	0.27	0.71	**0.77**
blow	48	783	197	0.00	**0.38**	0.30	0.20	**0.61**	0.58
break	43	826	205	0.00	**0.46**	0.29	0.13	**0.71**	0.54
breathe	16	458	116	0.03	**0.49**	0.31	0.44	**0.69**	0.66
call	15	685	171	0.02	0.56	0.56	0.35	0.60	**0.80**
cross	20	613	154	0.02	**0.49**	0.33	0.49	**0.66**	0.63
cry	12	794	198	0.05	0.49	0.49	0.60	0.55	**0.81**
enlarge	3	413	102	0.29	0.41	**0.85**	0.88	0.74	**0.94**
explain	5	389	97	0.12	**0.84**	0.67	0.59	0.82	**0.88**
forge	7	367	92	0.05	0.32	**0.37**	0.35	**0.50**	0.48
import	4	747	187	0.24	0.37	0.37	0.95	0.80	0.95
laugh	8	935	233	0.08	0.47	**0.78**	0.64	0.42	**0.88**
object	3	719	180	0.30	0.76	**0.85**	0.89	0.67	**0.97**
rush	9	609	152	0.06	0.36	**0.53**	0.55	0.67	**0.79**
say	3	366	91	0.30	0.63	**0.65**	0.91	0.86	**0.93**
sleep	11	941	236	0.07	**0.55**	0.51	0.81	0.63	**0.91**
smile	8	532	132	0.10	**0.53**	0.49	0.79	0.46	**0.86**
smoke	4	652	163	0.25	0.13	**0.50**	**0.98**	0.52	0.97
speed	8	483	122	0.10	0.16	**0.22**	0.77	0.84	**0.89**
talk	13	739	184	0.05	**0.35**	0.25	0.71	0.39	**0.81**
throw	32	398	104	0.01	**0.59**	0.32	0.19	**0.75**	0.64
wake	4	630	157	0.13	0.50	**0.82**	0.52	0.76	**0.89**
AVERAGE	11.96	650.6	162.84	0.12	0.50	**0.52**	0.62	0.68	**0.81**

Table 4: Results for the pattern disambiguation task
A is the baseline, B is the bootstrapping model, and C is the SVM model

is weighted with an association score based on the set of extracted triples, as described in equation 6. The thesaurus is the output of the application of a distance measure described in equation 7 between words w_i sharing similar tuples $||r, w'||$ (Rychlý and Kilgarriff, 2007).

The boostrapping algorithm uses the scores from the thesaurus to predict a label for each token. For each verb token v of the test set, it compares its contexts (r, w') to the contexts (r, w) labelled as k in the training set. The score for each token results from the sum of the contexts having the best score as described in equation 8.

Two thresholds have been tested, *minscore*, the minimal score returned by the algorithm, and *mindiff*, the minimal difference between the best score and the second best score.

3.5. Results

Table 4 shows the results obtained for both the bootstrapping system and the SVM classifier on the 25 verbs (selecting only normal examples from the corpus) in terms of both macro-average and micro-average F1. When a system gets the best score, it is shown in bold face. SVM (C) is clearly the best system overall verbs in terms of micro-average reaching 0.81 on average across the task. The baseline remains unbeaten for two verbs (for which one pattern almost completely covers the set of patterns), *smoke* (A=0.98) and *arouse* (A=0.95), with a difference of only 1 point in F1 from SVM (respectively 0.97 and 0.94). The inverse case also exists, where SVM beats the baseline from a short point; this is the case for *arrest* and *say*, with a draw for *import*.

The bootstrapping system (B=0.68) is slightly better than the baseline (A=0.62), even if it does not beat the baseline on half of the verbs (12/25). The best combination for this system is *mindiff* = 0.01, thus a low difference between the first two scores returned by the algorithm, and *minscore* = 0.9, thus a high score threshold. It is also worth noting that it is the best model on for seven verbs (arm, blow, break, breathe, cross, forge, throw), although the scores hardly reach 0.75. Among those verbs are those having a high number of patterns (over 20), which suggest that the bootstrapping approach scales better than SVM as the tagset grows.

This seems to be confirmed when we consider macro-average scores, which are much lower, and where SVM and bootstrapping systems achieve similar performance on average (0.52 and 0.50). The degradation of performance observed for both models indicates that the bootstrapping system is not as much affected by the skewed distribution as the SVM is. To finish, these systems do not achieve identical scores on each verb but can be considered, to some extent, complementary.

4. Error analysis

Detailed error analysis revealed that the main reason for the discrepancy between macro- and micro- averages is data size. Especially it is the size of training for each pattern which causes systems to score 0 in F1 on a verb pattern. This concerns about 40% of the patterns for both systems (115 for BS and 126 for SVM). The average number of examples for zero-scored patterns is 4.6 for BS and 2.9 for SVM. This means that the macro-scores would be higher if the systems were only evaluated on patterns for which there exists at least, say, 10 occurences of training data.

Figure 2 provides an additional view into how data size influence systems' performance. It represents the macro-average on portions of the data, separated according to the number of examples in total for each pattern. The chart shows three things. First, half of the patterns are tagged with 10 or less examples in total ($0 < x < 5$ and $5 < x < 10$ on the graph). Second, the curves show that F1 improves as the number of examples augments; this is more relevant for SVM than for BS. Third, on average, BS performs much better on patterns with 20 or less examples, while SVM performs better for patterns with more than 20 examples. This suggests that combining those two systems is worth considering, e.g. by using ensemble learning methods for model selection.

None of the system really uses semantic features: SVM Named Entities is as far as it gets in terms of semantics, and distributional similarity cannot be said to be semantics strictly speaking. However distributional similarity has proven to be beneficial, since it improved pattern disambiguation on several occasions. One of the reasons for errors in the bootstrapping model is its high dependency on the grammar, which has a lower coverage than the stanford dependency parser. This directly affects the bootstrapping model, which has a higher precision than recall (see Table 5).

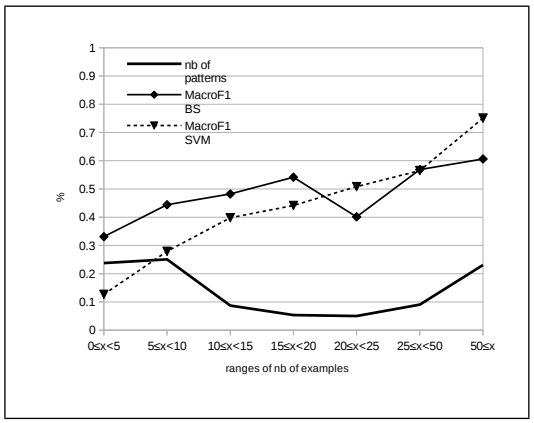

Figure 2: F1 scores according to data size

In addition, the SkE grammar, on which the bootstrapping approach is based, filters out pronouns, which could be a crucial discriminating feature between transitive and intransitive uses of a verb. Therefore, possible improvements will come from using a dependency parser to produce word sketches and integrate it in the bootstrapping approach. Finally, we believe that performing ontological groupings, e.g. using the CPA ontology, or SUMO attributes as in (Popescu, 2013), is worth investigating, since some patterns from the same verb share identical syntactic structure and only vary in terms of the semantic type of an argument.

Conclusions and perspectives

This article has focused on the task of pattern disambiguation in the context of the Pattern Dictionary of English Verbs. Two models have been tested on a set of 25 highly frequent verbs covering more than 20,000 tagged sentences. The first is a SVM implementation making use of the Stanford parser dependencies. The second is a bootstrapping approach based on a distributional thesaurus, available in the Sketch Engine.

Results obtained on the first model were satisfying in terms of micro-average F1, with the SVM reaching 0.81. However scores drop significantly when considering macro-average, which suggested that systems could be biased by the skewed distribution of labels. In-depth study of the data however revealed that the main reason for this drop is the low number of training examples for a significant number of patterns.

These results also need to be confirmed on an analysis of a larger set of verbs but are comparable to the ones obtained in WSD tasks. Although more experiments need to be performed, we hope that they will stimulate the use of PDEV in NLP applications where WSD is used, such as statistical machine translation.

5. Acknowledgements

This work was supported by AHRC grant [DVC, AH/J005940/1, 2012-2015] and by the Ministry of Education of Czech Republic within the LINDAT-Clarin project LM2010013.

model	micro-average			macro-average		
	precision	recall	f1	precision	recall	f1
SVM	0.81	0.81	0.81	0.54	0.51	0.52
BS	0.86	0.57	0.68	0.61	0.47	0.50

Table 5: Results in terms of precision and recall for the pattern disambiguation task

6. References

Carpuat, M. and Wu, D. (2007). How phrase sense disambiguation outperforms word sense disambiguation for statistical machine translation. In *11th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI 2007)*, Skovde.

De Marneffe, M.-C., MacCartney, B., Manning, C. D., et al. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

Finkel, J. R., Grenager, T., and Manning, C. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

Grefenstette, G. (1994). *Explorations in Automatic Thesaurus Discovery (The Springer International Series in Engineering and Computer Science)*. Springer.

Hanks, P. (2004). Corpus pattern analysis. In *Euralex Proceedings*, volume 1, pages 87–98.

Hanks, P. (2013). *Lexical Analysis: Norms and Exploitations*. MIT Press.

Kilgarriff, A. and Rychly, P. (2010). Semi-automatic dictionary drafting. In de Schryver, G.-M., editor, *Oxford Handbook of Innovation*. Menha Publichers, Kampala.

Kilgarriff, A., Rychly, P., Smrz, P., and Tugwell, D. (2004). The sketch engine. *Information Technology*, 105:116.

Lee, Y. K. and Ng, H. T. (2002). An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, EMNLP '02, pages 41–48, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lin, D. (1998). Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on Computational linguistics-Volume 2*. Association for Computational Linguistics.

Navigli, R. (2009). Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2):10:1–10:69.

Popescu, O. (2013). Learning corpus pattern with finite state automata. In *ICSC 2013 Proceedings*.

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

Rychlý, P. and Kilgarriff, A. (2007). An efficient algorithm for building a distributional thesaurus (and other sketch engine developments). In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 41–44, Prague, Czech Republic. Association for Computational Linguistics.

Toutanova, K. and Manning, C. D. (2000). Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13*, pages 63–70. Association for Computational Linguistics.