Distributional Semantics and CPA Pattern Disambiguation

First Author
Affiliation / Address line 1
email@domain

Second Author
Affiliation / Address line 1
email@domain

Abstract

This paper presents a methodology to automatically generate prototype vectors (PV) for patterns using the Pattern Dictionary of English Verbs (PDEV) repository, by means of a distributional semantic approach. In particular, we have generated, for each verb pattern in PDEV a PV by incorporating vectors (trained on Skip-gram Negative Sampling) of context words in patterns and concordances. We demonstrate how to use the resulting prototype vectors to disambiguate patterns in a computationally inexpensive way. In addition, we develop a modified model, called *Skip-gram Backward-Forward (SG-BF)*, which improves the overall performance of the disambiguation system.

1 Introduction

Word sense disambiguation (WSD) and word sense induction (WSI) have been the focus of research in Natural Language Processing for several years. To this end, there have been many efforts to create a framework for sense inventory creation and sense-annotated corpora. Among the most notable recent approaches is Corpus Pattern Analysis (CPA) (Hanks and Pustejovsky, 2005), with the output being the Pattern Dictionary of English Verbs (PDEV)[1], which has a rich inventory of patterns for over 1,000 English verbs. The corpus is a significant resource in its careful selection of patterns based on analyzing verb arguments and collocates in a corpus-driven manner. Given that the pattern library is still under development, however, there has been little opportunity to utilize it for downstream NLP experiments and applications.

Distributional semantic models (DSMs), based upon the assumption that words appearing in similar contexts are semantically related (Harris, 1968), have proven to be a robust and inexpensive approach for a number of applications, including WSD (McCarthy et al., 2004). In the past few years, some DSMs have been developed that successfully represent word vectors in low-dimensional space, such as the Skip-gram (SG) model, proposed by Mikolov et al. (2013b). Their model creates vectors that can be used effectively in analogy tasks (Mikolov et al., 2013a), e.g., as in the 2×2 matrix filling task: $king - man + woman \sim queen$. The success of the model results in part from the fact that it requires only a short training cycle, thereby providing a ready-to-use tool for analysis.

While the SG model has been successful for identifying some word relations, in particular named entity pairs such as *(Capital_City, Nation)*, some modifications must be made in order to adopt the model to capture verb semantic pattern behavior. For nouns, it is fairly straightforward to identify a fixed (and small) set of semantic relations among them, such as hyponymy and synonymy. Semantic relations for verbs are much subtler and therefore are harder to define. This is largely because of the complex nature of relational terms, whose meanings are modulated through composition and context (cf. (Mitchell and Lapata, 2008) and (Kintsch, 2001)).

In this paper, we examine the application of the Skip-gram model in WSD, using the PDEV pattern set and accompanying corpus (Hanks, 2013). In particular, we create, for each pattern selected from the PDEV corpus, a prototype vector, which will be used for disambiguation. We also devise a new model we call *Skip-gram Backward-forward (SG-BF)*, to be used specifically for verb pattern classification. Our experimental results show that while SG works well for the task of pattern disambiguation, SG-BF significantly improved the overall F1-score of the system.

[1]http://pdev.org.uk/

2 Related Work

This work is motivated both by recent research on DSMs that have proven useful to a number of NLP problems, such as syntactic parsing (Socher et al., 2013), named entity resolution (Passos et al., 2014), compositional semantics (Baroni et al., 2014), as well as by the philosophy behind Corpus Pattern Analysis (CPA) and its PDEV corpus (Hanks, 2013). We believe CPA and PDEV will be of great use for research on WSD and WSI, as initiated by the recent SemEval 2015-CPA task (Baisa et al., 2015).

Distributional semantic Models As mentioned above, distributional semantics constitutes a growing subfield within Computational Linguistics, having a large array of methods and approaches (cf. Clark, 2015). Recent approach includes Distributional Memory (Baroni and Lenci, 2010), a structured DSM taking into account the syntactic dependency of words. We adopt and simplify this idea in our work by using the relative location of the context word with the target word as clues for its syntactic relation. Mikolov et al. (2013b) develop the specific Skip-gram model that we adapt, which will be described in more detail below. There is also the work of Huang et al. (2012) and Neelakantan et al. (2014), among others, focusing on DSMs that generate multiple vectors for different senses of words. These approaches are neural network models that use clustering methods to learn multiple word-sense vectors, at the same time as training the network. Both groups have experimented with a fixed number of word sense vectors for each word, while the latter has also experimented with a variable number of word senses. We extend this idea to develop multiple prototype vectors for word patterns.

Work on Corpus Pattern Analysis The Semeval 2015-CPA task introduced some of the first open computational investigations using Corpus Pattern Analysis. The task mostly focuses on automatic derivation of verb patterns from a corpus. For PDEV pattern disambiguation, on the other hand, to the best of our knowledge, the only works reported are those performed by El Maarouf and Baisa (2013) and El Maarouf et al. (2014). Their systems will be explained in more detail when we discuss the results of our system. Both have been tested on a subset of 25 verbs and their corresponding verb patterns.

3 Background

3.1 PDEV Patterns and Corpus

The PDEV corpus is built upon the BNC, using the methodology of *Corpus Pattern Analysis*, first outlined in Pustejovsky et al. (2004), and then developed in Hanks and Pustejovsky (2005) and Hanks (2013). Corpus Pattern Analysis (CPA) is a technique for mapping meaning onto words in text. It is the methodology used to build the *Pattern Dictionary of English Verbs* (PDEV), which will be a fundamental resource for use in computational linguistics, language teaching, and cognitive science. It is based on the Theory of Norms and Exploitations (TNE, see Hanks 2004 and 2013, Hanks and Pustejovsky 2005, Pustejovsky et al.,2004). TNE in turn incorporates many ideas from Generative Lexicon (Pustejovsky, 1995), preference semantics (Yorick Wilks, 1975), and Sinclair's influential work on corpus analytics (eg. Sinclair 1966, 1987, 1991, 2004).

With CPA, verb senses are distinguished through corpus-derived syntagmatic patterns. A CPA pattern extends the traditional notion of selectional context to include a number of contextual features, such as minor category parsing, subphrasal cues, and a shallow semantic ontology. Accurate identification of the semantically relevant aspects of a pattern is not an obvious and straightforward procedure, as has sometimes been assumed in the literature. For example, the presence or absence of an adverbial of manner in the third valency slot around a verb can dramatically alter the meaning of a verb.

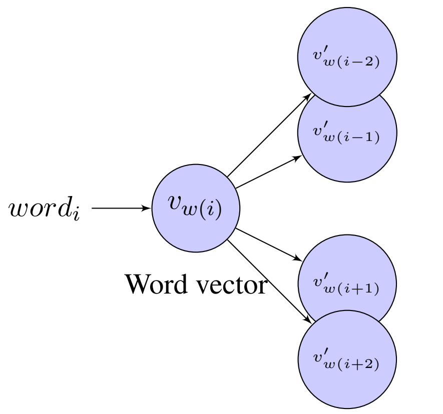
CPA patterns capture multiple facets of a word's usage. For each pattern, there is a partially defined structural template, providing the semantic categories of the collocate expressions (arguments). Additionally, implicatures can be associated with specific patterns. Take the verb *fire*, for example:

No.	Pattern
1	Pattern **Human** *fires* **Firearm** (**Direction**) Implicature **Human** causes **Firearm** to discharge a projectile towards **Physical_Object = Target**
3	Pattern **Human** or **Firearm** *fires* (**Direction**) Implicature **Human** causes **Firearm** to discharge a projectile (in a particular direction)

Table 1: Examples of patterns in PDEV

The PDEV corpus has a semantic ontology including semantic types such as *Human*, *Institution* etc. (Pustejovsky et al., 2006; Rumshisky et al., 2006). It should also be noted that PDEV makes a distinction in verb alternations, as reflected in the two patterns of the verb *fire* in Table 1: Pattern No. 3 is the no-object alternation of Pattern No. 1.

3.2 Skip-gram Negative Sampling Model



Context vector

Figure 1: **Skip-gram** Snapshot of training process at word $w(i)$, 4 context word vectors in its window size of 2.

In their paper, Mikolov et al. (2013b) experimented with a number of methods for learning word embedding vectors, including CBOW, skip-gram with negative sampling (SG-NS), and skip-gram with hierarchical softmax (SG-HS). The SG-NS model performed the best among these methods. The training time and memory usage of SG-NS is also lower because of its "on-the-fly" nature.

Let $w(i)$ be the word at position i of a document. In a Skip-gram model, there are two vectors corresponding to $w(i)$, $v_{w(i)}, v'_{w(i)} \in R^d$, where d is the number of embedded dimensions; v is the word vector, v' is the context vector. The purpose of the training method is to optimize the probability of seeing a word $w(k)$ in the context of another word $w(i)$. This probability can be represented as a softmax of the dot product between a target word vector ($v_{w(i)}$) and a context word vector ($v'_{w(k)}$), where k takes in the values of $\{i - window, .., i-1\} \cup \{i+1, .., i+window\}$:

$$P(w(k)|w(i)) = f(v'_{w(k)}, v_{w(i)}) = \frac{exp(v_{w(i)} \cdot v'_{w(k)})}{\sum_{u \in Vocabulary} exp(v_{w(i)} \cdot v'_u)}$$

However, a direct calculation of the softmax function and its differential involves iterating over the whole vocabulary, which is very expensive. An alternative is to learn a target function that can distinguish among samples observed from data and ones drawn from a noise distribution. The objective function therefore is modified to:

$$log(\frac{1}{1+exp(-v_{w(i)} \cdot v'_{w(k)})}) + \sum_{u \sim P_N(w)} log(1 - \frac{1}{1+exp(-v_{w(i)} \cdot v'_u)})$$

where $P_N(w)$ is a noise distribution that is scaled with the word distribution in the training corpus. The number of noise samples drawn from the noise distribution is a predetermined number. Optimizing the objective function is achieved by a method of stochastic gradient descent.

4 Skip-gram Negative Sampling Backward-Forward

The Skip-gram negative sampling Backward-Forward model (SG-NS-BF) takes into account the relative position of context word and current (target) word. This modification is motivated by the fact that the distribution of words before and after a verb typically correspond to its subject and complement collocations, respectively. Therefore, the context distribution before and after the verb are significantly different. A robust system applied for verb meaning needs to leverage this difference to achieve higher performance. While we do not use dependency parsing to obtain argument and complement features, using before/after relative positions of context word is already an improvement over the bag-of-word method, given the observation that for most of the verb patterns, one of two voices is significantly dominant over the other.

To capture the difference between the words before and after a verb, we modified the vector model of the skip-gram so that each vector is composed of two component vectors. For both the target word vector, v, and the context word vector, v', their left and right component vectors, which we will use $_{,L}$ and $_{,R}$ for denotation respectively, represent the distribution of words before and after them. The combined vector is made by stacking the left and right components together.

Learning in the SG-NS-BF model targets matching the left component of the current word vector and the right component of the context

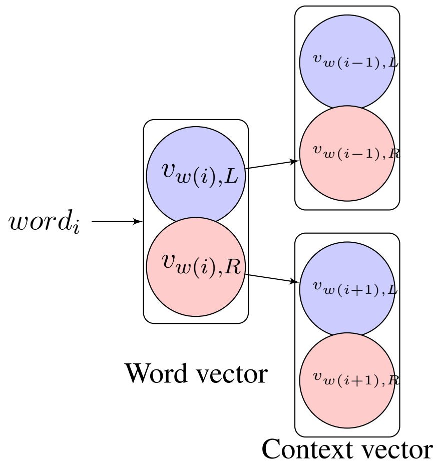


Figure 2: **SG-NS-BF** Snapshot of training process at word vector $v_{w(i)}$, 2 context word vectors in its window size of 1. The left component of word vector is associated with right component of context vector before it and vice versa.

word vector, if the context word appears before the current word and vice versa. The objective function for negative sampling therefore is changed to:

$$log(\frac{1}{1+exp(-v_{w(i),L}\cdot v'_{w(k),R})}) + \sum_{u\sim P_N(w)} log(1-\frac{1}{1+exp(-v_{w(i),L}\cdot v'_{u,R})})$$

if $k \in \{i - window, .., i - 1\}$

$$log(\frac{1}{1+exp(-v_{w(i),R}\cdot v'_{w(k),L})}) + \sum_{u\sim P_N(w)} log(1-\frac{1}{1+exp(-v_{w(i),R}\cdot v'_{u,L})})$$

if $k \in \{i + 1, .., i + window\}$

From now on, we use the terms SG-NS and SG (and likewise, SG-NS-BF and SG-BF) interchangeably, since we only experimented with the Negative Sampling approach of the Skip-gram model.

5 Evaluation

5.1 PDEV Verb and Pattern Selection

We used only verbs that have at least two senses that are statistically significant for our training purpose. In particular, any sense having fewer than 5 samples in the corpus was removed from the pattern corpus. Afterwards, we discarded all verbs having only one sense. Samples from concordances of the remaining verb-senses were split into a 80/20 distribution for training and testing purposes, respectively. Some general statistics for the selected verbs are reported in Table 2.

Total verbs	1276
Ambiguous verbs	688
Number of patterns	3074
Average pattern/verb	4.5
Training samples	166,288
Testing samples	43,101
Training/Testing	80/20

Table 2: Overall statistics of verbs used for training/testing

We also give statistics for certain verbs, most of them having more than 10 different patterns in Table 3. Among the remaining verbs, *blow* has the highest number of senses at 48.

5.2 Training and Testing Setup for PDEV Disambiguation

To evaluate our system, we used the Wikipedia corpus, based on a dump from March 31, 2014[2]. The corpus has more than 3 million documents and over 2 billion tokens. Some simple preprocessing steps were performed. First we removed markup tags, then we tokenized words by space, keeping only alphabetic words and lower cased them using the *gensim* Python library[3]. We POS-tagged the corpus using *spaCy* library in Python [4]. Part-of-speech of a word is attached to the word itself, so *America* will be changed to *america-n*. For both the SG and SG-BF models, we only kept the 200,000 most frequent words in the vocabulary (they shared the same vocabulary). It should be noted that we did not remove the high frequency stop words from the vocabulary, if they are function words, such as *on* and *in*: the PDEV corpus emphasizes prepositions as an important feature for distinguishing patterns.

Our model learned vectors of 300 dimensions, with the initial learning rate set at 0.025. The maximum context window was set at 4. This value is informed by the work on Skip-gram hyperparameter tuning of Levy et al. (2015). As reported in (Mikolov et al., 2013b), increasing

[2]http://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2. It was released on 2015/03/04.
[3]http://radimrehurek.com/gensim/about.html
[4]https://honnibal.github.io/spaCy/index.html

Verbs	Patterns		Samples			Percentage used	
	Used	Total	Total	Training	Testing	Pattern	Sample
scratch	9	14	193	138	41	64%	93 %
see	11	21	507	384	100	52%	95%
sleep	10	11	1235	982	250	91%	100%
appear	10	12	3845	3069	769	83%	100%
nurse	6	10	199	150	41	60%	96%
pour	18	20	658	516	137	90%	99%
blow	48	62	1338	1019	283	77%	97%
lock	15	16	792	626	163	94%	97%
plant	10	13	418	325	87	77%	99%
ask	7	12	995	784	199	58%	99%

Table 3: Selected verbs and their number of selected senses and samples in corpus

the number of iterations does improve the performance of word-analogy tasks, but not to a significant degree, so we only set 1 training iteration, since the time taken to train one iteration can be quite long (over 10 hours for each).

We use the following denotations for calculating verb-pattern prototype vectors. Let p be a pattern of a verb w that has definition D and concordance C of m samples. For each sample $s \in C$, let T_s be the context window of size z; $T_{s,L}$, $T_{s,R}$ respectively be z context words to the left and to the right of w; $T_s = T_{s,L} \cup T_{s,R}$. Prototype vector P is calculated as followings:

- **SG**

$$P_p = m * v_w + \sum_{s \in C, u \in T_s} v_u + \alpha * \sum_{u \in D} v_u$$

p's definition is treated just as a sample in the concordance, scaled with weight α.

- **SG-BF**

$$\begin{aligned} P_{p,L} &= m * v_{w,L} + \sum_{s \in C, u \in T_{s,L}} v_{u,R} \\ &+ \alpha * \sum_{u \in D} v_{u,R} \\ P_{p,R} &= m * v_{w,R} + \sum_{s \in C, u \in T_{s,R}} v_{u,L} \\ &+ \alpha * \sum_{u \in D} v_{u,L} \\ P_p &= P_{p,L} || P_{p,R} \end{aligned}$$

It is noted that for SG-BF, we incorporate into P's left component the right components of words in $T_{s,L}$. The reason for that peculiarity is for the context word $u \in T_{s,L}$, $v_{u,R}$ estimates the distribution of words after u, which better approximates context around target word w.

For evaluation, we use the window size of 4, empirically chosen from values $\{3, 4, 5\}$ and the value of α is set to $\frac{m}{5}$.

It should be pointed out that our verb-pattern prototype vectors do not differentiate inflectional forms of the verb, and they are grouped into the same set of prototype vectors: this is mainly because the size of the training data is not large enough to distinguish between different inflectional forms.

After the prototype vectors have been calculated, one can use them to disambiguate a verb in a new sample by calculating the context vector of the sample in the same manner as used for creating prototype vectors. Disambiguation is carried out by selecting the pattern prototype vector of the target verb that has highest cosine similarity with the calculated sample context vector.

5.3 Baseline

We took into consideration two baseline systems.

- **Most frequent label (MF)**: We used the same baseline as Maarouf and Baisa (2013). For each verb, the most frequent pattern is assigned to all test. This baseline, however, is too weak for the task, as many of the verbs have numerous patterns.

- **Random vector (RV)**: To evaluate the quality of word vectors themselves, we also used a second baseline that generate random word vectors and applied the SG averaging formula to calculate prototypes. The RV baseline is a better one to compare between SG and SG-BF.

5.4 Evaluation Metrics

For each verb, we calculate the average F1 score with several different methods. The *micro-*

average F1 score is the harmonic mean of the micro average precision and micro average recall, which are obtained by summing and dividing the error types for all patterns at once. While the micro-average F1 score is not a good indicator of system performance when there is a dominant pattern for a verb, the *macro average F1 score* can compensate for the micro average F1 score weakness by averaging the F1 score of each pattern for the same verb. See for example (Ghamrawi and McCallum, 2005). A *weighted* F1 score is also calculated. It is a variation of *macro* score, weighting patterns by their numbers of samples. It somewhat balances between *micro* and *macro* F1 score, and will be the main calculation method we use to evaluate our different models. In addition, we calculate the average F1 score across verbs weighted by number of samples in corpus[5].

6 Results

6.1 PDEV Pattern Disambiguation Results

We gave the results of our two different Skip-gram models applied for PDEV disambiguation against a simple baseline using most frequency pattern in the training data in Table 4. The result has showed that results of both model has gained substantial improvement against both baselines, and the Skip-gram Backward-forward model designed specifically for verbs also beats its general-used variant on most of considered verbs and on average F1-score (+2 gain for all average metrics).

We also compare the performance of our system with the results from El Maarouf et al. (2014). They proposed two systems that we will call Bootstrapping (BOOT) and Support Vector Machine (SVD). The former proposed a bootstrapping method that extracts collocates with verbs through dependency relation, and recognize shared features of verb patterns across training and testing data set. The later proposed to use dependency parsing to extract heads and dependents of verbs and learn using Support Vector Machine. They calculated unweighted average F1 score and experimented with only 25 verbs that are more frequently seen in the corpus, therefore we also calculate unweighted average F1 of our best system for 23 of their 25 verbs (taking out two verbs *break* and *rush* which at the point time of this paper, is not completed yet), and reported together with their two systems in Table 5 [6]:

	Boot	SVD	SG-BF
Average-Micro	68%	**82%**	75%
Average-Macro	50%	52%	**56%**

Table 5: Unweighted F1 score of (El Maarouf et al., 2014) two systems Bootstrapping (Boot) and Support Vector Machine (SVD) compared to SG-BF.

While SVD method beats our system in Micro-Average metric, SG-BF showed its strength in disambiguating skewed verbs (ones that have dominant pattern), reflecting by higher Macro-Average value. Moreover, our system doesn't require dependency parsing and in fact it could benefit from employing more sophisticated pre-processing as well.

6.2 SG-BF in Analogy Tasks

To evaluate the performance of the Backward-Forward model itself, we also used the analogy task defined by (Mikolov et al., 2013a), specifically for two syntactic categories *gram-7-past-tense* and *gram-9-plural-verbs*. In addition, we include one more test set we have created for verb antonym pairs which has analogy of type *import - export + open = close*. The result is reported in Table 6

Category	SG-NS	SG-NS-BF
gram-7-past-tense	46%	**54%**
gram-9-plural-verbs	61%	**83%**
antonym-verbs	11%	**18%**

Table 6: Performance of SG and SG-BF on some semantic-syntactic analogy tasks.

The improvement of SG-NS-BF model on all considered verb analogy tasks has proved that taking into account relative position of context and current word could provide better verb vectors.

6.3 Error Analysis

To provide a qualitative analysis of the error, we examined the verb *visit* as an example. It has three patterns listed in the PDEV corpus (Table 7). The

[5]We used weighted averaging instead of unweighted averaging as in Maarouf et al. (2014) because we experimented with all released verbs (688 verbs), while they only selected 25 frequent verbs. Performance of the different systems, therefore, needs to take into account the verb frequency parameter.

[6]Their calculation is on an older snapshot of PDEV.

Sel. verb	Micro-average F1				Macro-average F1				Weighted-average F1			
	MF	RV	SG	BF	MF	RV	SG	BF	MF	RV	SG	BF
scratch	20	32	**51**	46	4	29	**54**	41	6	31	**51**	44
see	30	47	55	**56**	4	30	38	**41**	14	47	55	**58**
sleep	80	57	69	**75**	9	40	**46**	38	71	63	73	**79**
appear	26	70	78	**86**	4	46	55	**58**	11	69	79	**86**
nurse	46	46	60	**61**	10	44	54	**58**	29	45	61	61
pour	31	42	**55**	48	3	37	**46**	44	15	41	**55**	48
blow	15	39	53	**55**	0	31	42	**43**	4	41	52	**55**
lock	20	51	52	**57**	2	32	38	**48**	7	53	52	**59**
plant	51	33	**56**	45	7	25	**35**	31	34	41	**59**	46
ask	40	72	65	**79**	8	65	55	**67**	23	75	65	**81**
All	53	59	70	**72**	16	44	56	**58**	39	61	71	**73**

Table 4: Performance of SG and SG-BF on PDEV disambiguation task across selected verbs and average. BF is short for SG-BF.

confusion matrix of the SG-BF results is given in Table 8:

No.	Pattern
1	**Human** or **Human_Group** *visits* **Location** **Human** goes to and spends some time in **Location** for tourism, business, or some other purpose
2	**Human 1** *visit* **Human 2** Human 1 goes to and spends some time with Human 2, typically for social reasons
3	**Human** *visit* (**Location**) Human goes to spend some time (typically only part of a day) at **Location = Educational** in order to see or learn something

Table 7: Pattern definition for the verb *visit*.

	1	2	3
1	143	24	20
2	4	77	0
3	2	0	3

Table 8: Confusion matrix of SG-BF result for the verb *visit*. Rows correspond to *true* labels, columns correspond to *predicted* labels.

As can be seen from the confusion matrix, most mistakes are from classifying instances of pattern 1 as pattern 2 or 3. Arguably, there is meaning overlapping between patterns 1 and 3, leading to a high confusion between these two classes. For the misclassification to pattern 2, there are two possible explanations. One is that the model failed to predict the semantic category of unseen human names, and indeed drops the objects of samples in pattern 2 out of the context window in training. Another more likely explanation is that collocates of type *Location* in the first pattern belong to different word distributional modes, such as general locations (*school, church, etc.*) and specific locations (*China, Washington etc*). Averaging over them might result in a low quality prototype vector. This suggests a model that would devise multiple prototype vectors for each pattern.

One more problem with the SG-BF model is that it seems to perform poorer than the SG model on verbs with less data. In particular, the weighted average F1 score went down for *plant* and *scratch*, while being the same for *nurse*. Another error that we noted from analyzing the verb *plant* is that it could not disambiguate the case where there is an alternation between a direct object and other oblique phrases (*plants a tree* vs *plants land with trees*). This problem is arguably inherent to both SG and SG-BF models. Further investigation into these errors will be left for future work.

7 Conclusion

In this paper, we have presented a method for applying distributional techniques, in particular, skip-gram models, to create verb pattern prototype vectors, in order to tackle the problem of pattern disambiguation in the PDEV corpus. We introduce a novel model that builds upon the skip-gram framework, taking into account relative positioning of word and context, and demonstrate that our model can build a set of high quality verb vectors, achieving a competitive performance on the task of PDEV pattern disambiguation, scoring the best Macro-Average F1 score among the current systems. Further, we believe this is the first experiment to evaluate and report results over the entire set of completed verbs from the PDEV corpus.

References

Vit Baisa, Jane Bradbury, Silvie Cinkova, Ismail El Maarouf, Adam Kilgarriff and Octavian Popescu. 2015. *SemEval-2015 Task 15: A Corpus Pattern Analysis Dictionary-Entry-Building Task*. SemEval 2015.

Marco Baroni and Alessandro Lenci. 2010. *Distributional Memory: A general framework for corpus-based semantics*. Computational Linguistics 36(4): 673-721.

Marco Baroni, Raffaella Bernardi, Roberto Zamparelli. 2014. *Frege in space: A program for compositional distributional semantics*. Linguistic Issues in Language Technologies 9(6): 5-110.

Stephen Clark. 2015. *Vector Space Models of Lexical Meaning*. Handbook of Contemporary Semantics second edition, edited by Shalom Lappin and Chris Fox. Wiley-Blackwell.

Ismail El Maarouf and Vit Baisa. 2013. *Automatic classification of patterns from the Pattern Dictionary of English Verbs*. Proceedings of JSSP.

Ismail El Maarouf, Jane Bradbury, Vit Baisa and Patrick Hanks. 2014. *Disambiguating Verbs by Collocation: Corpus Lexicography meets Natural Language Processing*. Proceedings of LREC, Reykjavik.

Christiane D. Fellbaum. 1998. *Wordnet: an electronic lexical database* MIT Press.

Nadia Ghamrawi and Andrew McCallum. 2005. *Collective multi-label classification*. In Proceedings of the 14th ACM international conference on Information and knowledge management (pp. 195-200). ACM.

Patrick Hanks and James Pustejovsky. 2005. *A pattern dictionary for natural language processing*. Revue Francaise de Linguistique Appliquee.

Patrick Hanks. 2013. *Lexical Analysis: Norms and Exploitations*, Cambridge. MIT Press.

Zellig S. Harris. 1968. *Mathematical Structures of Language.*, Wiley, New York.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. *Improving Word Representations via Global Context and Multiple Word Prototypes*. ACL.

Walter Kintsch. 2001. *Prediction*. Cognitive Science, 25(2), 173-202.

Omer Levy, Yoav Goldberg and Ido Dagan. 2015. *Improving Distributional Similarity with Lessons Learned from Word Embeddings*. TACL 2015.

Diana McCarthy, Rob Koeling, Julie Weeds, John Carroll. 2004. *Finding Predominant Word Senses in Untagged Text*, Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, 280287, Barcelona, Spain.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean *Efficient Estimation of Word Representations in Vector Space* In Proceedings of Workshop at ICLR 2013.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean *Distributed Representations of Words and Phrases and their Compositionality* In Proceedings of NIPS 2013.

Jeff Mitchell and Mirella Lapata. 2008. *Vector-based Models of Semantic Composition*. ACL.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, Andrew McCallum. 2014. *Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space*. EMNLP.

Alexandre Passos, Vineet Kumar, Andrew McCallum. 2014. *Lexicon Infused Phrase Embeddings for Named Entity Resolution*. CoNLL 2014.

James Pustejovsky 1995. *The Generative Lexicon*. MIT Press.

James Pustejovsky, Patrick Hanks and Anna Rumshisky 2004. *Automated Induction of Sense in Context*. COLING, Geneva, Switzerland.

James Pustejovsky, Catherine Havasi, Jessica Littman, Anna Rumshisky, and Marc Verhagen. 2006. *Towards a generative lexical resource: The brandeis semantic ontology*. Proceedings of the Fifth Language Resource and Evaluation Conference.

Anna Rumshisky, Patrick Hanks, Catherine Havasi, and James Pustejovsky. 2006. FLAIRS Conference.

John Sinclair. 1966. "Beginning the Study of Lexis" in C. E. Bazell, J. C. Catford, M. A. K. Halliday, and R. H. Robins (eds.) *In Memory of J. R. Firth*. Longman.

John Sinclair. 1987. *Looking Up: an Account of the Cobuild Project in Lexical Computing*. Harper-Collins.

John Sinclair. 1991. *Corpus, Concordance, Collocation*. Oxford University Press.

John Sinclair. 2004. *Trust the Text: Language, Corpus, and Discourse*. Routledge.

Richard Socher, John Bauer, Christopher D. Manning and Andrew Y. Ng. 2013. *Parsing with Compositional Vector Grammars*. In Proceedings of the ACL Conference 2013.

Yorick Wilks. 1975. *A Preferential, Pattern-Seeking, Semantics for Natural Language Inference*. In: Artificial Intelligence 6(1).