Dynamic and Static Prototype Vectors for Semantic Composition

Siva Reddy
University of York, UK
siva@cs.york.ac.uk

Ioannis P. Klapaftis
University of York, UK
giannis@cs.york.ac.uk

Diana McCarthy
Lexical Computing Ltd, UK
diana@dianamccarthy.co.uk

Suresh Manandhar
University of York, UK
suresh@cs.york.ac.uk

Abstract

Compositional Distributional Semantic methods model the distributional behavior of a compound word by exploiting the distributional behavior of its constituent words. In this setting, a constituent word is typically represented by a feature vector conflating all the senses of that word. However, not all the senses of a constituent word are relevant when composing the semantics of the compound. In this paper, we present two different methods for selecting the relevant senses of constituent words. The first one is based on Word Sense Induction and creates a static multi prototype vectors representing the senses of a constituent word. The second creates a single dynamic prototype vector for each constituent word based on the distributional properties of the other constituents in the compound. We use these prototype vectors for composing the semantics of noun-noun compounds and evaluate on a compositionality-based similarity task. Our results show that: (1) selecting relevant senses of the constituent words leads to a better semantic composition of the compound, and (2) dynamic prototypes perform better than static prototypes.

1 Introduction

Vector Space Models of lexical semantics have become a standard framework for representing a word's meaning. Typically these methods (Schütze, 1998; Pado and Lapata, 2007; Erk and Padó, 2008) utilize a bag-of-words model or syntactic dependencies such as subject/verb, object/verb relations, so as to extract the features which serve as the dimensions of the vector space. Each word is then represented as a vector of the extracted features, where the frequency of co-occurrence of the word with each feature is used to calculate the vector component asociated with that feature. Figure 1 provides an example of two nouns assuming a bag-of-words model.

	vector dimensions					
	animal	buy	apartment	price	rent	kill
house	⟨ 30	60	90	55	45	10 ⟩
hunting	⟨ 90	15	12	20	33	90 ⟩

Figure 1: A hypothetical vector space model.

Compositional Distributional Semantic methods formalise the meaning of a phrase by applying a vector composition function on the vectors associated with its constituent words (Mitchell and Lapata, 2008; Widdows, 2008). For example, the result of *vector addition* to compose the semantics of *house hunting* from the vectors **house** and **hunting** is the vector $\langle 120,\ 75,\ 102,\ 75,\ 78,\ 100 \rangle$.

As can be observed the resulting vector does not reflect the correct meaning of the compound *house hunting* due to the presence of irrelevant co-occurrences such as *animal* or *kill*. These co-occurrences are relevant to one sense of *hunting*, i.e. (*the activity of hunting animals*), but not to the sense of *hunting* meant in *house hunting*, i.e. *the activity of looking thoroughly*. Given that *hunting* has been associated with a single prototype (vector) by conflating all of its senses, the application of a composition function $\oplus$ is likely to include irrelevant co-occurrences in $\mathbf{house} \oplus \mathbf{hunting}$.

A potential solution to this problem would involve the following steps:

1. build separate prototype vectors for each of the senses of *house* and *hunting*
2. select the relevant prototype vectors of *house* and *hunting* and then perform the semantic composition.

In this paper we present two methods (section 3) for carrying out the above steps on noun-noun compounds. The first one (section 3.1) applies Word Sense Induction (WSI) to identity different senses (also called static multi prototypes) of the constituent words of a compound noun and then applies composition by choosing the relevant senses. The second method (section 3.2) does not identify a fixed set of senses. Instead, it represents each constituent by a prototype vector which is built dynamically (also called as a dynamic prototype) by activating only those contexts considered to be relevant to the constituent in the presence of the other constituent, and then performs the composition on the dynamic prototypes. For performing composition, we use vector composition functions.

Our evaluation (section 5) on a task for rating similarity between noun-noun compound pairs shows: (1) sense disambiguation of constituents improves semantic composition and (2) dynamic prototypes are better than static multi prototypes for semantic composition.

2 Related work

Any distributional model that aims to describe language adequately needs to address the issue of compositionality. Many distributional composition functions have been proposed in order to estimate the semantics of compound words from the semantics of the constituent words. Mitchell and Lapata (2008) discussed and evaluated various composition functions for phrases consisting of two words. Among these, the simple additive (ADD) and simple multiplicative (MULT) functions are easy to implement and competitive with respect to existing sophisticated methods (Widdows, 2008; Vecchi et al., 2011).

Let us assume a target compound noun N that consists of two nouns n and n'. Bold letters represent their corresponding distributional vectors obtained from corpora. $\oplus(\mathbf{N})$ denotes the vector of N obtained by applying the composition function $\oplus$ on $\mathbf{n}$ and $\mathbf{n}'$. Real number v_i denote the i^{th} co-occurrence in $\mathbf{v}$. The functions ADD and MULT following Mitchell and Lapata (2008) are defined as follows:

$$\begin{aligned} &\text{ADD:} && \oplus(\mathbf{N}) = \alpha\,\mathbf{n} + \beta\,\mathbf{n}' \\ &&& \text{i.e. } \oplus(\mathbf{N})_i = \alpha\,\mathrm{n}_i + \beta\,\mathrm{n}'_i \\ &\text{MULT:} && \oplus(\mathbf{N}) = \mathbf{n}\mathbf{n}' \\ &&& \text{i.e. } \oplus(\mathbf{N})_i = \mathrm{n}_i \,.\, \mathrm{n}'_i \end{aligned} \tag{1}$$

where α and β are real numbers.

Relevant to our work is the work of Erk and Padó (2008) who utilize a structured vector space model. The prototype vector of a constituent word is initially built, and later refined by removing irrelevant co-occurrences with the help of the selectional preferences of other constituents. The refined vectors are then used for the semantic composition of the compound noun. The results are encouraging showing that polysemy is a problem in vector space models. Our approach differs to theirs in the way we represent meaning - we experiment with static multi prototypes and dynamic prototypes. Our vector space model is based on simple bag-of-words which does not require selectional preferences for sense disambiguation and can be applied to resource-poor languages.

There are several other researchers who tried to address polysemy for improving the performance of different tasks but not particularly to the task of semantic composition. Some of them are Navigli and Crisafulli (2010) for web search results clustering, Klapaftis and Manandhar (2010b) for taxonomy learning, Reisinger and Mooney (2010) for word similarity and Korkontzelos and Manandhar (2009) for compositionality detection. In all cases, the reported results demonstrate that handling polysemy lead to improved performance of the corresponding tasks. This motivates our research for handling polysemy for the task of semantic composition using two different methods described in the next section.

3 Sense Prototype Vectors for Semantic Composition

In this section we describe two approaches for building sense specific prototype vectors of constituent words in a noun-noun compound. The first approach performs WSI to build static multi prototype vectors. The other builds a single dynamic prototype vector for each constituent by activating only the relevant exemplars of the constituent

with respect to the other constituent. An exemplar is defined as a corpus instance of a target word.

These sense specific prototype vectors are then used for semantic composition. Let N be the compound noun with constituents n and n'. Our aim is to select the relevant senses of n and n'.

3.1 Static Multi Prototypes Based Sense Selection

In the first stage (section 3.1.1), a WSI method is applied to both n and n'. The outcome of this stage is a set of clusters (senses). Each of these clusters is associated with a prototype vector taking the centroid of the cluster. Following Reisinger and Mooney (2010) we use the terminology *multi prototype vectors* in the meaning of *sense clusters*. Let $S(n)$ (resp. $S(n')$) be the set of prototypes of n, where each $s_i^n \in S(n)$ denotes the i^{th} sense of the noun n. Since these prototypes of constituents are static and do not change when the compound changes we refer to them as *static multi prototypes*.

In the next stage (section 3.1.2), we calculate all the pairwise similarities between the clusters of n and n', so as to select a pair of clusters with the highest similarity. The selected clusters are then combined using a composition function, to produce a single vector representing the semantics of the target compound noun N.

3.1.1 Graph-based WSI

Word Sense Induction is the task of identifying the senses of a target word in a given text. We apply a graph-based sense induction method, which creates a graph of target word instances and then clusters that graph to induce the senses. We follow the work of Klapaftis and Manandhar (2010a) for creating the graph and apply *Chinese Whispers* (CW) (Biemann, 2006), a linear graph clustering method that automatically identifies the number of clusters.

Figure 2 provides a running example of the different stages of the WSI method. In the example, the target word *mouse* appears with the *electronic device* sense in the contexts *A*, *C*, and with the *animal* sense in the contexts *B* and *D*.

Corpus preprocessing: Let bc denote the base corpus consisting of the contexts containing the target word tw. In our work, a context is defined by a set of words in a window of size 100 around the target.

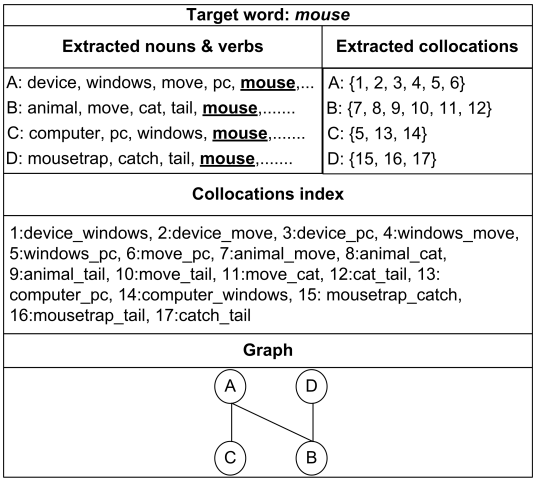

Figure 2: Running example of WSI

The aim of this stage is to capture words contextually related to tw. In the first step, the target word is removed from bc and part-of-speech tagging is applied to each context. Only nouns and verbs are kept and lemmatised. In the next step, the distribution of each word in the base corpus is compared to the distribution of the same noun in a reference corpus using the log-likelihood ratio (G^2) (Dunning, 1993). Words that have a G^2 below a pre-specified threshold (parameter p_1) are removed from each context of the base corpus. The result of this stage is shown in the upper left part of Figure 2.

Graph creation & clustering: Each context $c_i \in bc$ is represented as a vertex in a graph G. Edges between the vertices of the graph are drawn based on their similarity, defined in Equation 2, where $sm_{cl}(c_i, c_j)$ is the *collocational weight* of contexts c_i, c_j and $sm_{wd}(c_i, c_j)$ is their bag-of-words weight. If the edge weight $W(c_i, c_j)$ is above a prespecified threshold (parameter p_3), then an edge is drawn between the corresponding vertices in the graph.

$$W(c_i, c_j) = \frac{1}{2}(sm_{cl}(c_i, c_j) + sm_{wd}(c_i, c_j)) \quad (2)$$

Collocational weight: The limited polysemy of collocations is exploited to compute the similarity between contexts c_i and c_j. In this setting, a collocation is a juxtaposition of two words within the same context. Given a context c_i, a total of $\binom{N}{2}$ collocations are generated by combining each word with any other word in the context. Each collocation is weighted using the log-likelihood ratio (G^2) (Dunning, 1993) and is filtered out if the G^2

is below a prespecified threshold (parameter p_2). At the end of this process, each context c_i of tw is associated with a set of collocations (g_i) as shown in the upper right part of Figure 2 . Given two contexts c_i and c_j, the Jaccard coefficient is used to calculate the similarity between the collocational sets, i.e. $sm_{cl}(c_i, c_j) = \frac{|g_i \cap g_j|}{|g_i \cup g_j|}$.

Bag-of-words weight: Estimating context similarity using collocations may provide reliable estimates regarding the existence of an edge in the graph, however, it also suffers from data sparsity. For this reason, a bag-of-words model is also employed. Specifically, each context c_i is associated with a set of words (g_i) selected in the corpus preprocessing stage. The upper left part of Figure 2 shows the words associated with each context of our example. Given two contexts c_i and c_j, the bag-of-words weight is defined to be the Jaccard coefficient of the corresponding word sets, i.e. $sm_{wd}(c_i, c_j) = \frac{|g_i \cap g_j|}{|g_i \cup g_j|}$.

Finally, the collocational weight and bag-of-words weight are averaged to derive the edge weight between two contexts as defined in Equation 2. The resulting graph of our running example is shown on the bottom of Figure 2. This graph is the input to CW clustering algorithm. Initially, CW assigns all vertices to different classes. Each vertex i is processed for an x number of iterations and inherits the strongest class in its local neighborhood LN in an update step. LN is defined as the set of vertices which share a direct connection with vertex i. During the update step for a vertex i: each class C_k receives a score equal to the sum of the weights of edges (i, j), where j has been assigned class C_k. The maximum score determines the strongest class. In case of multiple strongest classes, one is chosen randomly. Classes are updated immediately, which means that a node can inherit classes from its LN that were introduced in the same iteration.

Experimental setting The parameters of the WSI method were fine-tuned on the nouns of the SemEval-2007 word sense induction task (Agirre and Soroa, 2007) under the second evaluation setting of that task, i.e. supervised (WSD) evaluation. We tried various parameter combinations shown in Table 1. Specifically, we selected the parameter combination p_1=15, p_2=10, p_3= 0.05 that maximized the performance in this evaluation. We use ukWaC (Ferraresi et al., 2008) corpus to retrieve all the instances of the target words.

Parameter	Range
G^2 word threshold (p_1)	15,25,35,45
G^2 collocation threshold (p_2)	10,15,20
Edge similarity threshold (p_3)	0.05,0.09,0.13

Table 1: WSI parameter values.

3.1.2 Cluster selection

The application of WSI on the nouns $n \in N$ and $n' \in N$ results in two sets of clusters (senses) $S(n)$ and $S(n')$. Cluster $S(n)$ is a set of contexts of the word n. Each context is represented as an exemplar $\mathbf{e}$, a vector specific to the context. Only the 10000 most frequent words in the ukWaC (along with their part-of-speech category) are treated as the valid co-occurrences i.e. the dimensionality of the vector space is 10000. For example, the exemplar of *hunting* in the context *"the-x purpose-n of-i autumn-n hunting-n be-v in-i part-n to-x cull-v the-x number-n of-i young-j autumn-n fox-n"* is ⟨ purpose-n:1, autumn-n:2, part-n:1, cull-v, number-n:1, young-j:1, fox-n:1 ⟩

For every cluster s_i^n in $S(n)$ we construct a prototype vector $\mathbf{v^{s_i^n}}$ by taking the centroid of all the exemplars in the cluster. Following Mitchell and Lapata (2008), the context words in the prototype vector are set to the ratio of probability of the context word given the target word to the overall probability of the context word[1].

The next step is to choose the relevant sense of each constituent for a given compound. We assume that the meaning of a compound noun can be approximated by identifying the most similar senses of each of its constituent nouns. Accordingly all the pairwise similarities between the $\mathbf{v^{s_i^n}}$ and $\mathbf{v^{s_i^{n'}}}$ are calculated using cosine similarity and the pair with maximum similarity is chosen for composition.

3.2 Dynamic Prototype Based Sense Selection

Kilgarriff (1997) argues that representing a word with a fixed set of senses is not a good way of modelling word senses. Instead word senses should be defined according to a given context. We propose a dynamic way of building word senses for the constituents of a given compound.

We use an exemplar-based approach to build the dynamic sense of a constituent with the help of other constituent. In exemplar-based modelling

[1]This is similar to pointwise mutual information without logarithm

followed by huge tarpon that like to use the	**light**	of your torch to help them hunt. At the
the Christmas trade this year or the	**lights**	will be off, probably for ever. The Merrymen
embrace better health - but doing so in the	**light**	of real and trusted information about the
present your organisation in a professional	**light**	and in a way our all our clients value.
continues to be significant, together with other	**light**	industries such as electrical engineering
and near-infrared light, along with red	**light**	emitted by hydrogen atoms and green light

Figure 3: Six random sentences of *light* from ukWaC

(Erk and Padó, 2010; Smith and Medin, 1981), each word is represented by all its exemplars without conflating them into a single vector. Depending upon the purpose, only relevant exemplars of the target word are activated. Exemplar-based models are more powerful than just prototype based ones because they retain specific instance information. As described in the previous section, an exemplar is a vector that represents a single instance of a given word in the corpus.

Let E_n be the set of exemplars of the word n. Given a compound N with constituents n and n', we remove irrelevant exemplars in E_n creating a refined set $E_n^{n'} \subset E_n$ with the help of the other constituent word n'. The prototype vector $\mathbf{n^{n'}}$ of n is then built from the centroid of the refined exemplar set $E_n^{n'}$. The vector $\mathbf{n^{n'}}$ represents the relevant prototype vector (sense) of n in the presence of the other constituent word n'. Unlike the static prototypes defined in the previous section, the prototype vectors of n and n' are built dynamically based on the given compound. Therefore, we refer to them as dynamic prototype vectors.

3.2.1 Building Dynamic Prototypes

We demonstrate our method of building dynamic prototypes with an example. Let us take the compound *traffic light*. Let $\mathbf{Traffic}$, $\mathbf{Light}$ and $\mathbf{TrafficLight}$ denote the prototype vectors of *traffic*, *light* and *traffic light* respectively. Word *light* occurs in many contexts such as quantum theory, optics, lamps and spiritual theory. In ukWaC, $light$ occurs with 316,126 exemplars. Figure 3 displays 6 random sentences of *light* from ukWaC. None of these exemplars are related to the target compound *traffic light*. When a prototype vector of *light* is built from all its exemplars, irrelevant exemplars add noise increasing the semantic differences between *traffic light* and *light* and thereby increasing the semantic differences between $\mathbf{TrafficLight}$ and $\mathbf{Traffic} \oplus \mathbf{Light}$. This is not desirable. The cosine similarity sim($\mathbf{Light}$, $\mathbf{TrafficLight}$) is found to be 0.27.

We aim to remove irrelevant exemplars of *light* with the help of the other constituent word *traffic* and then build a prototype vector of *light* which is related to the compound *traffic light*. Our intuition and motivation for exemplar removal is that it is beneficiary to choose only the exemplars of *light* which have context words related to *traffic* since the exemplars of *traffic light* will have context words related to both *traffic* and *light*. For example car, road, transport will generally be found within the contexts of all the words *traffic*, *light* and *traffic light*.

We rank each exemplar of *light* with the help of collocations of *traffic*. Collocations of *traffic* are defined as the context words which frequently occur with *traffic*, e.g. car, road etc. The exemplar of *light* representing the sentence "*Cameras capture **cars** running red **lights** ...*" will be ranked higher than the one which does not have context words related to *traffic*. We use Sketch Engine[2] (Kilgarriff et al., 2004) to retrieve the collocations of *traffic* from ukWaC. Sketch Engine computes the collocations using Dice metric (Dice, 1945). We build a collocation vector $\mathbf{Traffic^{colloc}}$ from the collocations of *traffic*.

We also rank each exemplar of *light* using the distributionally similar words to *traffic* i.e. words which are similar to *traffic* e.g. transport, flow etc. These distributionally similar words helps to reduce the impact of data sparseness and helps prioritize the contexts of *light* which are semantically related to *traffic*. Sketch Engine is again used to retrieve distributionally similar words of *traffic* from ukWaC. Sketch Engine ranks similar words using the method of Rychlý and Kilgarriff (2007). We build the vector $\mathbf{Traffic^{similar}}$ which consists of the similar words of *traffic*.

Every exemplar $\mathbf{e}$ from the exemplar set E_{light} [3] is finally ranked by

$$sim(\mathbf{e}, \mathbf{Traffic^{colloc}}) + sim(\mathbf{e}, \mathbf{Traffic^{similar}})$$

[2] Sketch Engine `http://www.sketchengine.co.uk`

[3] In E_{light}, we do not include the sentences which have the compound noun *traffic light* occurring in them.

We choose the top n% of the ranked exemplars in E_{light} to construct a refined exemplar set $E_{\text{light}}^{\text{traffic}}$. A prototype vector $\mathbf{Light}^{\mathbf{Traffic}}$ is then built by taking the centroid of $E_{\text{light}}^{\text{traffic}}$. $\mathbf{Light}^{\mathbf{Traffic}}$ denotes the sense of *light* in the presence of *traffic*. Since sense of *light* is built dynamically based on the given compound (here *traffic light*), we define $\mathbf{Light}^{\mathbf{Traffic}}$ as the dynamic prototype vector. The similarity sim($\mathbf{Light}^{\mathbf{Traffic}}$, $\mathbf{TrafficLight}$) is found to be 0.47 which is higher than the initial similarity 0.27 of $\mathbf{Light}$ and $\mathbf{TrafficLight}$. This shows that our new prototype vector of *light* is closer to the meaning of *traffic light*.

Similarly we build the dynamic prototype vector $\mathbf{Traffic}^{\mathbf{Light}}$ of *traffic* with the help of $light$. The dynamic prototypes $\mathbf{Traffic}^{\mathbf{Light}}$ and $\mathbf{Light}^{\mathbf{Traffic}}$ are used for semantic composition to construct $\mathbf{Traffic}^{\mathbf{Light}} \oplus \mathbf{Light}^{\mathbf{Traffic}}$

4 Composition functions

Given a compound, we perform composition using the sense based prototypes selected in the above section. We use the composition functions ADD and MULT described in Equation 1.

For the function ADD, we use equal weights for both constituent words i.e. $\alpha = \beta = 1$. For the function MULT there are no parameters.

5 Evaluation

Mitchell and Lapata (2010) introduced an evaluation scheme for semantic composition models. We evaluate on their dataset, describe the evaluation scheme, and present the results of various models.

5.1 Dataset

Mitchell and Lapata (2010) prepared a dataset[4] which contains pairs of compound nouns and their similarity judgments. The dataset consists of 108 compound noun pairs with each pair having 7 annotations from different annotators who judge the pair for similarity. A sample of 5 compound pairs is displayed in Table 2.

5.2 Evaluation Scheme

For each pair of the compound nouns, the mean value of all its annotations is taken to be the final similarity judgment of the compound.

[4]We would like to thank Jeff Mitchell and Mirella Lapata for sharing the dataset.

Annotator	N	N'	rating
4	phone call	committee meeting	2
25	phone call	committee meeting	7
11	football club	league match	6
11	health service	bus company	1
14	company director	assistant manager	7

Table 2: Evaluation dataset of Mitchell and Lapata (2010)

Let N and N' be a pair. To evaluate a model, we calculate the cosine similarity between the composed vectors $\oplus(\mathbf{N})$ and $\oplus(\mathbf{N}')$ obtained from the composition on sense based prototypes generated by the model. These similarity scores are correlated with human mean scores to judge the performance of the model.

5.3 Models Evaluated

We evaluate all the models w.r.t. the composition functions ADD and MULT.

Static Single Prototypes: This model does not perform any sense disambiguation and is similar to the method described in (Mitchell and Lapata, 2008). The prototype vector of each constituent formed by conflating all its instances is used to compose the vector of the compound.

Static Multi Prototypes: In the method described in section 3.1, word sense induction produces a large number of clusters i.e. static multi prototypes. We tried various parameters like choosing the target prototype of a constituent only from the top 5 or 10 large clusters.

Dynamic Prototypes: In the method described in section 3.2, the dynamic prototype of a constituent is produced from the top n% exemplars of the ranked exemplar set of the constituent. We tried various percent activation (n%) values - 2%, 5%, 10%, 20%, 50%, 80%.

Compound Prototype: We directly use the corpus instances of a compound to build the prototype vector of the compound. This method does not involve any composition. Ideally, one expects this model to give the best performance.

Static Multi Prototypes with Guided Selection: This is similar to Static Multi Prototypes model except in the way we choose the relevant prototype for each constituent. In section 3.1.2 we described an unsupervised way of prototype selection from multi prototypes. Unlike there, here we choose the constituent prototype (sense) which has the highest similarity to the prototype vector of the

Parameter Description	ADD	MULT
Static Single Prototypes		
	0.5173	0.6104
Static Multi Prototypes		
Top 5 clusters	0.1171	0.4150
Top 10 clusters	0.0663	0.2655
Dynamic Prototypes		
Top 2 % exemplars	0.6261	**0.6552**
Top 5 % exemplars	0.6326	0.6478
Top 10 % exemplars	**0.6402**	0.6515
Top 20 % exemplars	0.6273	0.6359
Top 50 % exemplars	0.5948	0.6340
Top 80 % exemplars	0.5612	0.6355
Static Multi Prototypes with Guided Selection		
Top 5 clusters	0.2290	0.4187
Top 10 clusters	0.2710	0.4140
Compound Prototype		
0.4152		

Table 3: Spearman Correlations of Model predictions with Human Predictions

compound. This is a guided way of sense selection since we are using the compound prototype vector which is built from the compound's corpus instances. The performance of this model gives us an idea of the upper boundary of multi prototype models for semantic composition.

5.4 Results and Discussion

All the above models are evaluated on the dataset described in section 5.1. Table 3 displays the Spearman correlations of all these models with the human annotations (mean values).

The results of Static Single Prototypes model are consistent with the previous findings of Mitchell and Lapata (2010), in which MULT performed better than ADD.

All the parameter settings of Dynamic Prototypes outperformed Static Single Prototypes. This shows that selecting the relevant sense prototypes of the constituents improve semantic composition. We also observe that the highest correlation is achieved by including just the top 2% exemplars for each constituent. It seems that as the sample of exemplars increases, noise increases as well, and this results in a worse performance.

The comparison between Static Single Prototypes and Static Multi Prototypes shows that the former performs significantly better than the latter. This is not according to our expectation. The possible reason for poor performance could be because of the sense selection process (section 3.1.2) which might have failed to choose the relevant sense of each constituent word.

However, Static Multi Prototypes with Guided Sense Selection still fail to perform better than Static Single Prototypes. Therefore, we can conclude that the lower performance of Static Multi Prototypes cannot be attributed to the sense selection process only. Despite that, the applied graph clustering method results in the generation of a very large number of clusters, some of which refer to the same word usage with subtle differences. Hence, our future work focuses on a selection process that chooses multiple relevant clusters of a constituent word. Additionally, our ongoing work suggests that the use of verbs as features in the graph creation process (section 3.1.1) causes the inclusion of noisy edges and results in worse clustering.

Our evaluation also shows that Dynamic Prototypes provide a better semantic composition than Static Multi Prototypes. The main reason for this result stems from the fact that Dynamic Prototypes explicitly identify the relevant usages of a constituent word with respect to the other constituent and vice versa, without having to deal with a set of issues that affect the performance of Static Multi Prototypes such as the clustering and the sense se-

lection process.

The performance of Compound Prototype is lower than the compositional models. The reason could be due to the data sparsity. Data sparsity is known to be a major problem for modelling the meaning of compounds. In a way, the results are encouraging for compositional models.

In all these models, the composition function MULT gave a better performance than ADD.

6 Conclusions

This paper presented two methods for dealing with polysemy when modeling the semantics of a noun-noun compound. The first one represents senses by creating static multi prototype vectors, while the second represents context-specific sense of a word by generating a dynamic prototype vector. Our experimental results show that: (1) sense disambiguation improves semantic composition, and (2) dynamic prototypes are a better representation of senses than static multi prototypes for the task of semantic composition.

In future, we would like to explore other static multi prototype approaches of Reisinger and Mooney (2010) and Klapaftis and Manandhar (2010a) in comparison with dynamic prototypes. Dynamic prototypes are found to be particularly encouraging since they present a different mechanism for sense representation unlike traditional methods.

Acknowledgements

The authors are grateful to Lexical Computing Ltd for providing a free installation of Sketch Engine at University of York. The authors would like to thank anonymous reviewers for their excellent feedback. This work is supported by the European Commission via the EU FP7 INDECT project, Grant No.218086, Research area: SEC-2007-1.2-01 Intelligent Urban Environment Observation System.

References

Eneko Agirre and Aitor Soroa. 2007. Semeval-2007 task 02: Evaluating word sense induction and discrimination systems. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 7–12, Prague, Czech Republic, June. Association for Computational Linguistics.

Chris Biemann. 2006. Chinese Whispers - An Efficient Graph Clustering Algorithm and its Application to Natural Language Processing Problems. In *Proceedings of TextGraphs*, pages 73–80, New York, USA.

Lee R. Dice. 1945. Measures of the amount of ecologic association between species. *Ecology*, 26(3):pp. 297–302.

Ted Dunning. 1993. Accurate Methods for the Statistics of Surprise and Coincidence. *Computational Linguistics*, 19(1):61–74.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 897–906, Stroudsburg, PA, USA. Association for Computational Linguistics.

Katrin Erk and Sebastian Padó. 2010. Exemplar-based models for word meaning in context. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 92–97, Stroudsburg, PA, USA. Association for Computational Linguistics.

Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. 2008. Introducing and evaluating ukWaC, a very large web-derived corpus of english. In *Proceedings of the WAC4 Workshop at LREC 2008*, Marrakesh, Morocco.

Adam Kilgarriff, Pavel Rychly, Pavel Smrz, and David Tugwell. 2004. The sketch engine. In *Proceedings of EURALEX 2004*.

Adam Kilgarriff. 1997. I don't believe in word senses. In *Computers and the Humanities, 31(2):91-113.*

Ioannis Klapaftis and Suresh Manandhar. 2010a. Word sense induction & disambiguation using hierarchical random graphs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 745–755, Cambridge, MA, October. Association for Computational Linguistics.

Ioannis P. Klapaftis and Suresh Manandhar. 2010b. Taxonomy Learning Using Word Sense Induction. In *Proceedings of NAACL-HLT-2010*, pages 82–90, Los Angeles, California, June. ACL.

Ioannis Korkontzelos and Suresh Manandhar. 2009. Detecting compositionality in multi-word expressions. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, ACLShort '09, pages 65–68, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pages 236–244, Columbus, Ohio.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*.

Roberto Navigli and Giuseppe Crisafulli. 2010. Inducing word senses to improve web search result clustering. In *EMNLP*, pages 116–126.

Sebastian Pado and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Joseph Reisinger and Raymond J. Mooney. 2010. Multi-prototype vector-space models of word meaning. In *HLT-NAACL*, pages 109–117.

Pavel Rychlý and Adam Kilgarriff. 2007. An efficient algorithm for building a distributional thesaurus (and other sketch engine developments). In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 41–44, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hinrich Schütze. 1998. Automatic Word Sense Discrimination. *Computational Linguistics*, 24(1):97–123.

Edward E. Smith and Douglas L. Medin. 1981. *Categories and concepts / Edward E. Smith and Douglas L. Medin*. Harvard University Press, Cambridge, Mass. :.

Eva Maria Vecchi, Marco Baroni, and Roberto Zamparelli. 2011. (linear) maps of the impossible: Capturing semantic anomalies in distributional space. In *Proceedings of the Workshop on Distributional Semantics and Compositionality*, pages 1–9, Portland, Oregon, USA, June. Association for Computational Linguistics.

Dominic Widdows. 2008. Semantic vector products: Some initial investigations. In *Second AAAI Symposium on Quantum Interaction, Oxford*, March.