A comparison of models of word meaning in context

Georgiana Dinu
Universität des Saarlandes
Saarbrücken, Germany
dinu@coli.uni-saarland.de

Stefan Thater
Universität des Saarlandes
Saarbrücken, Germany
stth@coli.uni-saarland.de

Sören Laue
Friedrich-Schiller Universität
Jena, Germany
soeren.laue@uni-jena.de

Abstract

This paper compares a number of recently proposed models for computing context sensitive word similarity. We clarify the connections between these models, simplify their formulation and evaluate them in a unified setting. We show that the models are essentially equivalent if syntactic information is ignored, and that the substantial performance differences previously reported disappear to a large extent when these simplified variants are evaluated under identical conditions. Furthermore, our reformulation allows for the design of a straightforward and fast implementation.

1 Introduction

The computation of *semantic similarity* scores between words is an important sub-task for a variety of NLP applications (Turney and Pantel, 2010). One standard approach is to exploit the so-called *distributional hypothesis* that similar words tend to appear in similar contexts: Word meaning is represented by the contexts in which a word occurs, and semantic similarity is computed by comparing these contexts in a high-dimensional vector space.

Such distributional models of word meaning are attractive because they are simple, have wide coverage, and can be easily acquired in an unsupervised way. Ambiguity, however, is a fundamental problem: when encountering a word in context, we want a distributional representation which reflects its meaning in this specific context. For instance, while *buy* and *acquire* are similar when we consider them in isolation, they do not convey the same meaning when *acquire* occurs in *students acquire knowledge.* This is particularly difficult for vector space models which compute a single *type vector* summing up over all occurrences of a word. This vector mixes all of a word's usages and makes no distinctions between its—potentially very diverse—senses.

Several proposals have been made in the recent literature to address this problem. Type-based methods combine the (type) vector of the target with the vectors of the surrounding context words to obtain a disambiguated representation. In recent work, this has been proposed by Mitchell and Lapata (2008), Erk and Padó (2008) and Thater et al. (2010; 2011), which differ in the choice of input vector representation and in the combination operation they propose.

A different approach has been taken by Erk and Padó (2010), Reisinger and Mooney (2010) and Reddy et al. (2011), who make use of *token vectors* for individual *occurrences* of a word, rather than using the already mixed type vectors. Generally speaking, these methods "select" a set of token vectors of the target, which are similar to the current context, and use only these to obtain a disambiguated representation.

Yet another approach has been taken by Dinu and Lapata (2010), Ó Séaghdha and Korhonen (2011) and Van de Cruys et al. (2011), who propose to use latent variable models. Conceptually, this comes close to token-based models, however their approach is more unitary as they attempt to recover a hidden layer which best explains the observation data.

In this paper, we focus on the first group of approaches and investigate the precise differences between the three models of Erk and Padó and Thater et al., out of which (Thater et al., 2011) achieves state of the art results on a standard data set. Despite the fact that these models exploit similar intuitions, both their formal presentations and the results obtained vary to a great extent. The answer given in this paper is surprising: the three models are essentially equivalent if syntactic information is ignored; in a syntactic space the three methods implement only slightly different intuitions. We clarify these connections, simplify the syntactic variants originally proposed and reduce them to straightforward matrix operations, and evaluate them in a unified experimental setting. We obtain significantly better results than originally reported in the literature. Our reformulation also also supports efficient implementations for these methods.

2 Models for meaning in context

We consider the following problem: we are given an *occurrence* of a target word and want to obtain a vector that reflects its meaning in the given context. To simplify the presentation, we restrict ourselves to contexts consisting of a single word, and use *acquire* in context *knowledge* as a running example.

EP08. Erk and Padó (2008) compute a contextualized vector for *acquire* by combining its type vector ($\vec{w}$) with the inverse selectional preference vector of *knowledge* (c). This is simply the centroid of the vectors of all words that take *knowledge* as direct object (r):

$$v(w,r,c) = \left(\frac{1}{n} \sum_{w'} f(w',r,c) \cdot \vec{w'} \right) \times \vec{w} \qquad (1)$$

where $f(w',r,c)$ denotes the co-occurrence association between the context word c and words w' related to c by grammatical relation r in a training corpus; n is the number of words w' and $\times$ denotes a vector composition operation. In this paper, we take $\times$ to be point-wise multiplication, which is reported to work best in many studies in the literature.

TFP10. Thater et al. (2010) also compute contextualized vectors by combing the vectors of the target word and of its context. In contrast to EP08, however, they use second order vectors as basic representation for the target word.

$$\vec{w} = \sum_{r,r',w''} \left(\sum_{w'} f(w,r,w') \cdot f(w',r',w'') \right) \vec{e}_{r,r',w''} \quad (2)$$

That is, the vector for a target word w has components for all combinations of two grammatical roles r, r' and a context word w'; the inner sum gives the value for each component.

The contextualized vector for *acquire* is obtained through pointwise multiplication with the (1st-order) vector for *knowledge* ($\vec{c}$), which has to be "lifted" first to make the two vectors comparable:

$$v(w,r,c) = \vec{w} \times L_r(\vec{c}) \qquad (3)$$

$\vec{c} = \sum_{r',w'} f(c,r',w')\vec{e}_{(r',w')}$ is a first order vector for the context word; the "lifting map" $L_r(\vec{c})$ maps this vector to $\sum_{r',w'} f(c,r',w')\vec{e}_{(r,r',w')}$ to make it compatible with $\vec{w}$.

TFP11. Thater et al. (2011) take a slightly different perspective on contextualization. Instead of combing vector representations for the target word and its context directly, they propose to re-weight the vector components of the target word, based on distributional similarity with the context word:

$$v(w,r,c) = \sum_{r',w'} \alpha(r,c,r',w') \cdot f(w,r',w') \cdot \vec{e}_{(r',w')} \quad (4)$$

where $\alpha(r,c,r',w')$ is simply $cos(\vec{c},\vec{w'})$ if r and r' denote the same grammatical function, else 0.

3 Comparison

The models presented above have a number of things in common: they all use syntactic information and "second order" vectors to represent word meaning in context. Yet, their formal presentations differ substantially. We now show that the models are essentially equivalent *if we ignore syntax*: they component-wise multiply the second order vector of one word (target or context) with the first order vector of the other word. Specifically, we obtain the following derivations, where $W = \{w_1, ..., w_n\}$ denotes the vocabulary, and V the symmetric $n \times n$ input matrix, where $V_{ij} = f(w_i, w_j)$ gives the co-occurrence association between words w_i and w_j:

$$\begin{aligned} v_{\text{EP08}}(w,c) &= \frac{1}{n} \sum_{w'} \left(f(w',c) \cdot \vec{w'} \right) \times \vec{w} \\ &= \frac{1}{n} \sum_{w'} \left(f(w',c) \cdot \langle f(w',w_1), \ldots \rangle \right) \times \vec{w} \\ &= \frac{1}{n} \langle \sum_{w'} f(w',c) \cdot f(w',w_1), \ldots \rangle \times \vec{w} \\ &= \frac{1}{n} \langle <\vec{c},\vec{w}_1>, \ldots, <\vec{c},\vec{w}_n> \rangle \times \vec{w} \\ &= \frac{1}{n} \vec{c}\, V \times \vec{w} \end{aligned}$$

$$
\begin{aligned}
v_{\mathrm{TFP10}}(w,c) &= \sum_{w''\in W}\left(\sum_{w'\in W} f(w,w')\cdot f(w',w'')\right)\vec{e}_{w''}\times\vec{c}\\
&= \langle \sum_{w'\in W} f(w,w')f(w',w_1),\ldots\rangle\times\vec{c}\\
&= \langle <\vec{w},\vec{w}_1>,\ldots,<\vec{w},\vec{w}_n>\rangle\times\vec{c}\\
&= \vec{w}\,V\times\vec{c}
\end{aligned}
$$

$$
\begin{aligned}
v_{\mathrm{TFP11}}(w,c) &= \sum_{w'\in W}\alpha(c,w')\cdot f(w,w')\cdot\vec{e}_{w'}\\
&= \langle\alpha(w_1,c)\cdot f(w,w_1),\ldots\rangle\\
&= \langle\alpha(w_1,c),\ldots\rangle\times\vec{w} && (*)\\
&= \langle<\vec{w}_1,\vec{c}>,\ldots,<\vec{w}_n,\vec{c}>\rangle\times\vec{w}\\
&= \vec{c}\,V\times\vec{w}
\end{aligned}
$$

where $<\vec{v},\vec{w}>$ denotes scalar product. In step (*), we assume that $\alpha(w,c)$ denotes the scalar product of $\vec{w}$ and $\vec{c}$, instead of cosine similarity, as TFP11. This is justified if we assume that all vectors are normalized, in which case the two are identical.

As it can be observed the syntax-free variants of EP08 and TFP11 are identical up to the choice in normalization. TFP10 proposes an identical model to that of TFP11, however with a different interpretation, in which the roles of the context word and of the target word are swapped.

4 Evaluation

We have just shown that EP08, TFP10 and TFP11 are essentially equivalent to each other if syntactic information is ignored, hence it is a bit surprising that performance results reported in the literature vary to such a great extent. In this section we consider syntactic variants of these methods and we show that performance differences previously reported can only partly be explained by the different ways syntactic information is used: when we simplify these models and evaluate them under identical conditions, the differences between them disappear to a large extent.

To evaluate the three models, we reimplemented them using matrix operations similar to the ones used in Section 3, where we made few simplifications to the TFP10 and EP08 models: we follow TFP11 and we use component-wise multiplication to combine the target with one context word, and add the resulting composed vectors when given more context words[1]. Furthermore for TFP10, we change the treatment of syntax in the line of the much simpler proposal of TFP11. Specifically:

Model	GAP	Δ Literature
EP08	46.6	+ 14.4 (32.2)*
TFP10	48.3	+ 3.9 (44.4)
TFP11	51.8	±0.0
TFP10+11	52.1	N/A

Table 1: GAP scores LST data.
* The best available GAP score for this model (from Erk and Padó (2010)) is reported only on a subset of the data - this subset is however judged by the authors to be "easier" than the entire data; all other methods are tested on the entire dataset.

$$v(w,r,c) = L_{r^{-1}}(VV^T)_{w,:}\times V_{c,:} \qquad \text{(TFP10)}$$

$$v(w,r,c) = V_{w,:}\times L_r(VV^T)_{c,:} \qquad \text{(TFP11)}$$

where V is a $I\times J$ *syntactic* input matrix, i.e. the columns are (word, relation) pairs. For simplification, the columns of V are reordered such that syntactic relations form continuous regions. L_r is a lifting map similar to that of Equation (3) as it maps I- into J-dimensional vectors: the resulting vector is equal to the original one in the column region of relation r, while everything else is 0. In the above equations we use the standard Matlab notation, $V_{w,:}$ denoting a row vector in matrix V.

We evaluate these models on a paraphrase ranking task, using the SemEval 2007 Lexical Substitution Task (LST) dataset: the models are given a target word in context plus a list of potential synonyms (substitution candidates) ranging over all senses of the target word. The models have to decide to what extent each substitution candidate is a synonym of the target *in the given context.* We omit the precise description of the evaluation setting here, as we follow the methodology described in Thater et al. (2011).

Results are shown in Table 1, where the first column gives the GAP (Generalized Average Precision) score of the model and the second column gives the difference to the result reported in the literature. TFP10 and EP08 perform much better than the original proposals, as we obtain very significant gains of 4 and 14 GAP points.

[1]Note that some of the parameters in the EP08 method (omitted in the brief presentation in Section 2), which are difficult to tune (Erk and Padó (2009)), disappear this way.

We can observe that the differences between the three methods, when simplified and tested in an unified setting, largely disappear. This is to be expected as all three methods implement very similar, all motivated intuitions: TFP11 reweights the vector of the target *acquire* with the second order vector of the context *knowledge*, i.e. with the vector of similarities of *knowledge* to all other words in the vocabulary. TFP10 takes a complementary approach: it reweights the vector of *knowledge* with the second order vector of *acquire*. In both these methods, anything outside the *object* (*object*$^{-1}$ respectively) region of the space, is set to 0. The variant of EP08 that we implement is very similar to TFP11, however it compares *knowledge* to all other words in the vocabulary only using occurrences *as objects* while TFP11 takes *all* syntactic relations into account.

Note that TFP10 and TFP11 operate on complementary syntactic regions of the vectors. For this reason the two models can be trivially combined. The combined model (TFP10+11) achieves even better results: the difference to TFP11 is small, however statistically significant at level $p < 0.05$.

Implementation details. Straightforward implementations of the three models are computationally expensive, as they all use "second order" vectors to implement contextualization of a target word. Our reformulation in terms of matrix operations allows for efficient implementations, which take advantage of the sparsity of the input matrix V: contextualization of a target word runs in $O(nnz(V))$, where *nnz* is the number of non-zero entries. Note that ranking not only a small set of predefined substitution candidates, as in the experiment above, but also ranking the entire vocabulary runs in $O(nnz(V))$. On this task, this overall running time is in fact identical to that of simpler methods such as those of Mitchell and Lapata (2008).

In our experiments, we use GigaWord to extract a syntactic input matrix V of size $\approx 2M \times 7M$. V is only 4.5×10^{-06} dense. Note that because of the simple operations involved, we do not need to compute or store the entire VV^T matrix, which is much denser than V (we have estimated order of 10^{10} entries). The sparsity of V allows for very efficient computations in practice: the best single model, TFP11, runs in less than 0.2s/0.4s per LST instance, for ranking the candidate list/entire vocabulary in a Python implementation using scipy.sparse, on a standard 1GHz processor.

5 Conclusions

In this paper, we have compared three related vector space models of word meaning in context. We have reformulated the models and showed that they are in fact very similar. We also showed that the different performances reported in the literature are only to some extent due to the differences in the models: We evaluated simplified variants of these and obtained results which are (much) better than previously reported, bringing the three models much closer together in terms of performance. Aside from clarifying the precise relationship between the three models under consideration, our reformulation has the additional benefit of allowing the design of a straightforward and efficient implementation.

Finally, our focus on these methods is justified by their clear advantages over other classes of models: unlike token-based or latent variable methods, they are much simpler and require no parameter tuning. Furthermore, they also obtain state of the art results on the paraphrase ranking task, outperforming other simple type-based methods (see (Van de Cruys et al., 2011) and (Ó Séaghdha and Korhonen, 2011) for results of other methods on this data).

Acknowledgments. This work was partially supported by the Cluster of Excellence "Multimodal Computing and Interaction", funded by the German Excellence Initiative.

References

Georgiana Dinu and Mirella Lapata. 2010. Measuring distributional similarity in context. In *Proceedings of EMNLP 2010*, Cambridge, MA.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of EMNLP 2008*, Honolulu, HI, USA.

Katrin Erk and Sebastian Padó. 2009. Paraphrase assessment in structured vector space: Exploring parameters and datasets. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*.

Katrin Erk and Sebastian Padó. 2010. Exemplar-based models for word meaning in context. In *Proceedings of ACL 2010 Short Papers*, Uppsala, Sweden.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, Columbus, OH, USA.

Diarmuid Ó Séaghdha and Anna Korhonen. 2011. Probabilistic models of similarity in syntactic context. In *Proceedings of EMNLP 2011*.

Siva Reddy, Ioannis Klapaftis, Diana McCarthy, and Suresh Manandhar. 2011. Dynamic and static prototype vectors for semantic composition. In *Proc. of IJCNLP 2011*.

Joseph Reisinger and Raymond J. Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Proceedings of NAACL 2010*, Los Angeles, California.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2010. Contextualizing semantic representations using syntactically enriched vector models. In *Proceedings of ACL 2010*, Uppsala, Sweden.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2011. Word meaning in context: A simple and effective vector model. In *Proceedings of IJCNLP 2011*.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space modes of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Tim Van de Cruys, Thierry Poibeau, and Anna Korhonen. 2011. Latent vector weighting for word meaning in context. In *Proceedings of EMNLP 2011*.