LIMSI: Learning Semantic Similarity by Selecting Random Word Subsets

Artem Sokolov
LIMSI-CNRS
B.P. 133, 91403 Orsay, France
artem.sokolov@limsi.fr

Abstract

We propose a semantic similarity learning method based on Random Indexing (RI) and ranking with boosting. Unlike classical RI, we use only those context vector features that are informative for the semantics modeled. Despite ignoring text preprocessing and dispensing with semantic resources, the approach was ranked as high as 22nd among 89 participants in the SemEval-2012 Task6: Semantic Textual Similarity.

1 Introduction

One of the popular and flexible tools of semantics modeling are vector distributional representations of texts (also known as vector space models, semantic word spaces or distributed representations). The principle idea behind vector space models is to use word usage statistics in different contexts to generate a high-dimensional vector representations for each word. Words are represented by context vectors whose closeness in the vector space is postulated to reflect semantic similarity (Sahlgren, 2005). The approach rests upon the *distributional hypothesis*: words with similar meanings or functions tend to appear in similar contexts. The prominent examples of vector space models are Latent Semantic Analysis (or Indexing) (Landauer and Dutnais, 1997) and Random Indexing (Kanerva et al., 2000).

Because of the heuristic nature of distributional methods, they are often designed with a specific semantic relation in mind (synonymy, paraphrases, contradiction, etc.). This complicates their adaption to other application domains and tasks, requiring manual trial-and-error feature redesigns and tailored preprocessing steps to remove morphology/syntax variations that are not supposed to contribute to the semantics facet in question (e.g., stemming, stopwords). Further, assessing closeness of semantic vectors is usually based on a fixed simple similarity function between distributed representations (often, the cosine function). The cosine function implicitly assigns equal weights to each component of the semantic vectors regardless of its importance for the particular semantic relation and task. Finally, during production of training and evaluation sets, the continuum of possible grades of semantic similarity is usually substituted with several integer values, although often only the relative grade order matters and not their absolute values. Trying to reproduce the same values or the same gaps between grades when designing a semantic representation scheme may introduce an unnecessary bias.

In this paper we address all of the above drawbacks and present a semantic similarity learning method based on Random Indexing. It does not require manual feature design, and is automatically adapted to the specific semantic relations by selecting needed important features and/or learning necessary feature transformations before calculating similarity. In the proof-of-concept experiments on the SemEval-2012 data we deliberately ignored all routine preprocessing steps, that are often considered obligatory in semantic text processing, we did not use any of the semantic resources (like WordNet) nor trained different models for different data domains/types. Despite such over-constrained setting, the method showed very positive performance and

*First Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 543–546,
Montréal, Canada, June 7-8, 2012. ©2012 Association for Computational Linguistics

was ranked as high as 22nd among 89 participants.

2 Random Indexing

Random Indexing (RI) is an alternative to LSA-like models with large co-occurrence matrices and separate matrix decomposition phase to reduce dimension. RI constructs context vectors on-the-fly based on the occurrence of words in contexts. First, each word is assigned a unique and randomly generated high-dimensional sparse ternary vector. Vectors contain a small number (between 0.1-1%) of randomly distributed +1s and -1s, with the rest of the elements set to 0. Next, the final context vectors for words are produced by scanning through the text with a sliding window of fixed size, and each time the word occurs in the text, the generated vectors of all its neighbors in the sliding context window are added to the context vector of this word[1]. Finally, the obtained context vectors are normalized by the occurrence count of the word.

RI is a practical variant of the well-known dimension reduction technique of the Johnson-Lindenstrauss (JL) lemma (Dasgupta and Gupta, 2003). An Euclidean space can be projected with a random Gaussian matrix R onto smaller dimension Euclidean space, such that with high probability the distance between any pair of points in the new space is within a distortion factor of $1 \pm \varepsilon$ of their original distance. Same or similar guarantees also hold for a uniform $\{-1, +1\}$-valued or ternary (from a certain distribution) random R (Achlioptas, 2003) or for even sparser matrices (Dasgupta et al., 2010)

Restating the JL-lemma in the RI-terminology, one can think of the initial space of characteristic vectors of word sets of all contexts (each component counts corresponding words seen in the context window over the corpus) embedded into a smaller dimension space, and approximately preserving distances between characteristic vectors. Because of the ternary generation scheme, each resulting feature-vector dimension either rewards, penalizes or "switches off" certain words for which the corresponding row of R contained, resp., $+1$, -1 or 0.

So far, RI has been a naïve approach to feature learning – although it produces low-dimensional feature representations, it is unconscious of the learning task behind. There is no guarantee that the Euclidean distance (or cosine similarity) will correctly reflect the necessary semantic relation: for a pair of vectors, not all word subsets are characteristic of a particular semantic relation or specific to it, as presence or absence of certain words may play no role in assessing given similarity type. Implications of RI in the context of learning textual similarity are coming from the feature selection (equivalently, word subset selection) method, based on boosting, that selects only those features that are informative for the semantic relation being learned (Section 4). Thus, the supervision information on sentence similarity guides the choose of word subsets (among all randomly generated by the projection matrix) that happen to be relevant to the semantic annotations.

3 Semantic Textual Similarity Task

Let $\{(\boldsymbol{s}_1^i, \boldsymbol{s}_2^i)\}$ be the training set of N pairs of sentences, provided along with similarity labels y_i. The higher the value of y_i the more semantically similar is the pair $(\boldsymbol{s}_1^i, \boldsymbol{s}_2^i)$. Usually absolute values of y_i are chosen arbitrary; only their relative order matters.

We would learn semantic similarity between $(\boldsymbol{s}_1^i, \boldsymbol{s}_2^i)$ as a function $H(\bar{x}^i)$, where $\bar{x}^i$ is a single vector combining sentence context vectors $v(\boldsymbol{s}_1^i)$ and $v(\boldsymbol{s}_2^i)$. Context representation $v(\boldsymbol{s})$ for a sentence $\boldsymbol{s}$ is defined as an average of the word context vectors $v(w)$ contained in it, found using a large text corpus with the RI approach, described in the previous section: $v(\boldsymbol{s}) = \sum_{w \in \boldsymbol{s}} v(w) / |\boldsymbol{s}|$. Possible transformations into $\bar{x}^i$ include a concatenation of $v(\boldsymbol{s}_1^i)$ and $v(\boldsymbol{s}_2^i)$, concatenation of the sum and difference vectors or a vector composed of component-wise symmetric functions (e.g., a product of corresponding components). In order to learn a symmetric H, one can either use each pair twice during training, or symmetrize the construction of $\bar{x}$.

4 Feature Selection with Boosting

We propose to exploit natural ordering of $(\boldsymbol{s}_1^i, \boldsymbol{s}_2^i)$ according to y^i to learn a parameterized similarity function $H(\bar{x}^i)$. In this way we do not try learning the absolute values of similarity provided in the training. Also, by using boosting approach we allow

[1] Although decreasing discounts dampening contribution of far-located context words may by beneficial, we do not use it putting our method in more difficult conditions.

for gradual inclusion of features into similarity function H, implementing in this way feature selection.

For a given number of training steps T, a boosting ranking algorithm learns a scoring function H, which is a linear combination of T simple, non-linear functions h_t called weak learners: $H(\bar{x}) = \sum_{t=1}^{T} \alpha_t h_t(\bar{x})$, where each α_t is the weight assigned to h_t at step t of the learning process.

Usually the weak learner is defined on only few components of $\bar{x}$. Having build H at step t, the next in turn $(t+1)$'s leaner is selected, optimized and weighted with the corresponding coefficient α_{t+1}. In this way the learning process selects only those features in $\bar{x}$ (or, if viewed from the RI perspective, random word subsets) that contribute most to learning the desired type input similarity.

As the first ranking method we applied the pair-wise ranking algorithm RankBoost (Freund et al., 2003), that learns H by minimizing a convex approximation to a weighted pair-wise loss:

$$\sum_{(\boldsymbol{s}_1^i,\boldsymbol{s}_2^i),(\boldsymbol{s}_1^j,\boldsymbol{s}_2^j):y^i<y^j} P(i,j)[\![H(\bar{x}^i) \geq H(\bar{x}^j)]\!].$$

Operator $[\![A]\!] = 1$ if the $A =$ true and 0 otherwise. Positive values of P weight pairs of $\bar{x}^i$ and $\bar{x}^j$ – the higher is $P(i,j)$, the more important it is to preserve the relative ordering of $\bar{x}^i$ and $\bar{x}^j$. We used the simplest decision stumps that depend on one feature as weak learners: $h(\boldsymbol{x};\theta,k) = [\![x^k > \theta]\!]$, where k is a feature index and θ is a learned threshold.

The second ranking method we used was a point-wise ranking algorithm, based on gradient boosting regression for ranking (Zheng et al., 2007), called RtRank and implemented by Mohan et al. (2011)[2]. The loss optimized by RtRank is slightly different:

$$\sum_{(\boldsymbol{s}_1^i,\boldsymbol{s}_2^i),(\boldsymbol{s}_1^j,\boldsymbol{s}_2^j):y^i<y^j} (\max\{0, H(\bar{x}^i) - H(\bar{x}^j)\})^2.$$

Another difference is in the method for selecting weak learner at each boosting step, that relies on regression loss and not scalar product as RankBoost. Weak learners for RtRank were regression trees of fixed depth (4 in our experiments).

5 Experiments

We learned context vectors on the GigaWord English corpus. The only preprocessing of the corpus was stripping all tag data, removing punctuation and lowercasing. Stop-words were not removed. Context vectors were built with the JavaSDM package (Hassel, 2004)[3] of dimensionality $N = 100$ and $N = 10^5$, resp., for preliminary and final experiments, with random degree 10 (five +1s and -1s in each initial vector), right and left context window size of 4 words[4] and constant weighting scheme.

	learner	**transform**	**correl.**	**σ**
baseline	pure RI, cos	-	0.264	0.005
	logistic reg.	-	0.508	0.041
	logistic reg.	concat	0.537	0.052
boosting	RankBoost	sumdiff	0.685	0.027
		product	0.663	0.018
		crossprod	0.648	0.028
		crossdiff	0.643	0.023
		concat	0.625	0.025
		absdiff	0.602	0.021
	RtRank	sumdiff	0.730	0.020
		product	0.721	0.023

Table 1: Mean performance of the transformation and boosting methods for $N = 100$ on train data.

Training and test data provided in the SemEval-2012 Task 6 contained 5 training and 5 testing text sets each of different domains or types of sentences (short video descriptions, pairs of outputs of a machine translation system, etc.). Although the 5 sets had very different characteristics, we concatenated all training files and trained a single model. The principal evaluation metrics was Pearson correlation coefficient, that we report here. Two related other measures were also used (Agirre et al., 2012).

Obtained sentence vectors $v(\boldsymbol{s})$ for were transformed into vectors $\bar{x}$ with several methods:

- 'sumdiff': $\bar{x} = (\bar{v}(\boldsymbol{s}_1) + \bar{v}(\boldsymbol{s}_2), sgn(v_1(\boldsymbol{s}_1) - v_1(\boldsymbol{s}_2))(v(\boldsymbol{s}_1) - v(\boldsymbol{s}_2)))$
- 'concat': $\bar{x} = (v(\boldsymbol{s}_1), v(\boldsymbol{s}_2))$, and $\bar{x}' = (v(\boldsymbol{s}_2), v(\boldsymbol{s}_1))$
- 'product': $x_i = v_i(\boldsymbol{s}_1) \cdot v_i(\boldsymbol{s}_2)$
- 'crossprod': $x_{ij} = v_i(\boldsymbol{s}_1) \cdot v_j(\boldsymbol{s}_2)$
- 'crossdiff': $x_{ij} = v_i(\boldsymbol{s}_1) - v_j(\boldsymbol{s}_2)$
- 'absdiff': $x_i = |v_i(\boldsymbol{s}_1) - v_i(\boldsymbol{s}_2)|$.

Methods 'concat' and 'sumdiff' were proposed by Hertz et al. (2004) for distance learning for clus-

[2] http://sites.google.com/site/rtranking

[3] http://www.csc.kth.se/∼xmartin/java

[4] Little sensitivity was found to the window sizes from 3 to 6.

learner	**transform**	**train**$\pm\sigma$	**test**	**rank**	MSRpar	MSRvid	SMTeur	OnWN	SMTnews
RankBoost	product	0.748±0.017	0.6392	32	0.3948	0.6597	0.0143	0.4157	0.2889
	sumdiff	0.735±0.016	0.6196	45	0.4295	0.5724	0.2842	0.3989	0.2575
RtRank	product	0.784±0.017	**0.6789**	**22**	0.4848	0.6636	0.0934	0.3706	0.2455
	sumdiff	0.763±0.014							

Table 2: Mean performance of the best-performing two transformation and two boosting methods for $N = 10^5$.

tering. Comparison of mean performance of different transformation and learning methods on the 5-fold splitting of the training set is given in Table 1 for short context vectors ($N = 100$). The correlation is given for the optimal algorithms' parameters (T for RankBoost and, additionally, tree depth and random ratio for RtRank), found with cross-validation on 5 folds. With these results for small N, two transformation methods were preselected ('sumdiff' and 'product') for testing and submission with $N = 10^5$ (Table 2), as increasing N usually increased performance. Yet, only about 10^3 features were actually selected by RankBoost, meaning that a relatively few random word subsets were informative for approximating semantic textual similarity.

In result, RtRank showed better performance, most likely because of more powerful learners, that depend on several features (word subsets) simultaneously. Performance on machine translation test sets was the lowest that can be explained by very poor quality of the training data[5]: models for these subsets should have been trained separately.

6 Conclusion

We presented a semantic similarity learning approach that learns a similarity function specific to the semantic relation modeled and that selects only those word subsets in RI, presence of which in the compared sentences is indicative of their similarity, by using only relative order of the labels and not their absolute values. In spite of paying no attention to preprocessing, nor using semantic corpora, and with no domain adaptation the method showed promising results.

Acknowledgments

This work has been funded by OSEO under the Quaero program.

References

Dimitris Achlioptas. 2003. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *Comput. Syst. Sci.*, 66:671–687.

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proc. of the Int. Workshop on Semantic Evaluation (SemEval 2012) // Joint Conf. on Lexical & Computational Semantics (*SEM 2012)*.

Sanjoy Dasgupta and Anupam Gupta. 2003. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Struct. Algorithms*, 22(1):60–65.

Anirban Dasgupta, Ravi Kumar, and Tamás Sarlos. 2010. A sparse Johnson-Lindenstrauss transform. In *Proc. of the ACM Symp. on Theory of Comput.*, pages 341–350.

Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. 2003. An efficient boosting algorithm for combining preferences. *Mach.Learn.Res.*, 4:933–969.

Martin Hassel. 2004. JavaSDM - a Java package for working with Random Indexing and Granska.

Tomer Hertz, Aharon Bar-hillel, and Daphna Weinshall. 2004. Boosting margin based distance functions for clustering. In *ICML*, pages 393–400.

Pentti Kanerva, Jan Kristoferson, and Anders Holst. 2000. Random indexing of text samples for latent semantic analysis. In *Proc. of the Conf. of the Cogn. Science Society*.

Thomas K. Landauer and Susan T. Dutnais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychol. Rev.*, pages 211–240.

Ananth Mohan, Zheng Chen, and Kilian Q. Weinberger. 2011. Web-search ranking with initialized gradient boosted regression trees. *Mach.Learn.Res.*, 14:77–89.

Magnus Sahlgren. 2005. An introduction to random indexing. In *Workshop on Methods & Applic. of Sem. Indexing // Int. Conf. on Terminol. & Knowl. Eng.*

Zhaohui Zheng, Hongyuan Zha, Tong Zhang, Olivier Chapelle, Keke Chen, and Gordon Sun. 2007. A general boosting method and its application to learning ranking functions for web search. In *NIPS*.

[5] A reviewer suggested another reason: more varied or even incorrect lexical choice that is sometimes found in MT output.