Penn: Using Word Similarities to better Estimate Sentence Similarity

Sneha Jha and **H. Andrew Schwartz** and **Lyle H. Ungar**
University of Pennsylvania
Philadelphia, PA, USA
{jhasneha, hansens, ungar}@seas.upenn.edu

Abstract

We present the Penn system for SemEval-2012 Task 6, computing the degree of semantic equivalence between two sentences. We explore the contributions of different vector models for computing sentence and word similarity: Collobert and Weston embeddings as well as two novel approaches, namely *eigenwords* and *selectors*. These embeddings provide different measures of distributional similarity between words, and their contexts. We used regression to combine the different similarity measures, and found that each provides partially independent predictive signal above baseline models.

1 Introduction

We compute the semantic similarity between pairs of sentences by combining a set of similarity metrics at various levels of depth, from surface word similarity to similarities derived from vector models of word or sentence meaning. Regression is then used to determine optimal weightings of the different similarity measures. We use this setting to assess the contributions from several different word embeddings.

Our system is based on similarities computed using multiple sets of features: (a) naive lexical features, (b) similarity between vector representations of sentences, and (c) similarity between constituent words computed using WordNet, using the *eigenword* vector representations of words , and using *selectors*, which generalize words to a set of words that appear in the same context.

2 System Description

This section briefly describes the feature sets used to arrive at a similarity measure between sentences. We compare the use of word similarities based on three different embeddings for words neural embeddings using recursive autoencoders, *eigenwords* and *selectors*.

2.1 Neural Models of Word Representation

An increasingly popular approach is to learn representational embeddings for words from a large collection of unlabeled data (typically using a generative model), and to use these embeddings to augment the feature set of a supervised learner. These models are based on the distributional hypothesis in linguistics that words that occur in similar contexts tend to have similar meanings. The similarities between these vectors indicate similarity in the meanings of corresponding words.

The state of the art model in paraphrase detection uses an unsupervised recursive autoencoder (RAE) model based on an unfolding objective that learn feature vectors for phrases in syntactic parse trees (Socher et al., 2011). The idea of neural language models is to jointly learn an embedding of words into an n-dimensional vector space that capture distributional syntactic and semantic information via the words co-occurrence statistics. Further details and evaluations of these embeddings are discussed in Turian et al. (2010).

Once the distributional syntactic and semantic matrix is learned on an unlabeled corpus, one can use it for subsequent tasks by using each words vector to represent that word. For initial word embeddings, we used the 100-dimensional vectors com-

*First Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 679–683,
Montréal, Canada, June 7-8, 2012.

puted via the unsupervised method of Collobert and Weston (2008). These word embeddings are matrices of size $|V| \times n$ where $|V|$ is the size of the vocabulary and n is the dimensionality of the semantic space. This matrix usually captures co-occurrence statistics and its values are learned. We used the embeddings provided by Socher et al. (2011). Although the original paper employed a dynamic pooling layer in addition to the RAE that captures the global structure of the similarity matrix, we found the resulting sentence-level RAE itself was useful. In turn, we use these vector representations at the sentence level where the cosine similarity between the sentence vectors serves as a measure of sentence similarity. All parameters for the RAE layer are kept same as described by Socher et al. (2011).

2.2 Eigenword Similarity

Recent spectral methods use large amounts of unlabeled data to learn word representations, which can then be used as features in supervised learners for linguistic tasks. *Eigenwords*, a spectral method for computing word embeddings based on context words that characterize the meanings of words, can be efficiently computed by a set of methods based on singular value decomposition (Dhillon et al., 2011).

Such representations are dense, low dimensional and real-valued like the vector representations in the previous section except that they are induced using eigen-decomposition of the word co-occurrence matrix instead of neural networks. This method uses Canonical Correlation Analysis (CCA) between words and their immediate contexts to estimate word representations from unlabeled data. CCA is the analog to Principal Component Analysis (PCA) for pairs of matrices. It computes the directions of maximal correlation between a pair of matrices. CCAs invariance to linear data transformations enables proofs showing that keeping the dominant singular vectors faithfully captures any state information. (For this work, we used the Google n-gram collection of web three-grams as the unlabeled data.) Each dimension of these representations captures latent information about a combination of syntactic and semantic word properties. In the original paper, the word embeddings are context-specific. For this task, we only use context-oblivious embeddings i.e. one embedding per word type for this task, based on their model. Word similarity can then be calculated as cosine similarity between the eigenword representation vectors for any two words.

To move from word-level similarity to sentence-level a few more steps are necessary. We adapted the method of matrix similarity given by Stevenson and Greenwood (2005). One calculates similarity between all pairs of words, and each sentence is represented as a binary vector (with elements equal to 1 if a word is present and 0 otherwise). The similarity between these sentences vectors $\vec{a}$ and $\vec{b}$ is given by:

$$s(\vec{a}, \vec{b}) = \frac{\vec{a} W \vec{b}}{|\vec{a}||\vec{b}|} \quad (1)$$

where W is a semantic similarity matrix containing information about the similarity of word pairs. Each element in matrix W represents the similarity of words according to some lexical or spectral similarity measure.

2.3 Selector Similarity

Another novel method to account for the similarity between words is via comparison of Web selectors (Schwartz and Gomez, 2008). *Selectors* are words that take the place of an instance of a target word within its local context. For example, in "he addressed the strikers at the rally", selectors for 'strikers' might be 'crowd', 'audience', 'workers', or 'students' words which can realize the same constituent position as the target word. Since selectors are determined based on the context, a set of selectors is an abstraction for the context of a word instance. Thus, comparing selector sets produces a measure of word instance similarity. A key difference between selectors and the eigenwords used in this paper are that selectors are instance specific. This has the benefit that selectors can distinguish word senses, but the drawback that each word instance requires its own set of selectors to be acquired.

Although selectors have previously only been used for worse sense disambiguation, one can also use them to compute similarity between two word instances by taking the cosine similarity of vectors containing selectors for each instance. In our case, we compute the cosine similarity for each pair of noun instances and populate the semantic similarity matrix in formula (1) to generate a sentence-level

similarity estimate. Combining web selector- based word similarity features with the word embeddings from the neural model gave us the best overall performance on the aggregated view of the data sets.

2.4 Other Similarity Metrics

Knowledge-Based. We use WordNet to calculate semantic distances between all open-class words in the sentence pairs. There are three classifications of similarity metrics over WordNet: path-based, information- content based, and gloss-based (Pedersen et al., 2004). We chose to incorporate those measures performing best in the Schwartz & Gomez (2011) application-oriented evaluation: (a) the path-based measure of Schwartz & Gomez (2008); (b) the information-content measure of Jiang & Conrath (1997) utilizing the difference in information content between concepts and their point of intersection; (c) the gloss-based measure of Patwardhan & Pedersen (2006). By including metrics utilizing different sources of information, we suspect they will each have something novel to contribute.

Because WordNet provides similarity between concepts (word senses), we take the maximum similarity between all senses of each word to be the similarity between the two words. Such similarity can then be computed between multiple pairs of words to populate the semantic similarity matrix W in formula (1) and generate sentence-level similarity estimates as described above. The information-content and path-based measures are restricted to comparing nouns and verbs and only across the same part of speech. On the other hand, the gloss-based measure, which relies on connections through concept definitions, is more general and can compare words across parts of speech.

Surface Metrics. We added the following set of lexical features to incorporate some surface information lost in the vector-based representations.

- difference in the lengths of the two sentences
- average length of the sentences
- number of common words based on exact string match
- number of content words in common
- number of common words in base form
- number of similar numerals in the sentences

3 Evaluation and Results

We combine the similarity metrics discussed previously via regression (Pedregosa et al., 2011). We included the following sets of features:

- **System-baseline:** surface metrics, knowledge-based metrics. (discussed in section 2.4).
- **Neu:** Neural Model similarity (section 2.1)
- **Ew:** Eigenword similarity (section 2.2)
- **Sel:** Selector similarity (section 2.3)

To capture possible non-linear relations, we added a squared and square-rooted column corresponding to each feature in the feature matrix. We also tried to combine all the features to form composite measures by defining multiple interaction terms. Both these sets of additional features improved the performance of our regression model. We used all features to train both a linear regression model and a regularized model based on ridge regression. The regularization parameter for ridge regression was set via cross-validation over the training set. All predictions of similarity values were capped within the range [0,1]. Our systems were trained on the following data sets:

- MSR-Paraphrase, Microsoft Research Paraphrase Corpus-750 pairs of sentences.
- MSR-Video, Microsoft Research Video Description Corpus-750 pairs of sentences.
- SMT-Europarl, WMT2008 development data set (Europarl section)-734 pairs of sentences.

Our performance in the official submission for the SemEval task can be seen in Table 1. **LReg** indicates the run with linear regression, **ELReg** adds the eigenwords feature and **ERReg** also uses eigenwords but with ridge regression. At the time of submission, we were not ready to test with the selector features yet. Ridge regression consistently outperformed linear regression for every run of our system, but overall Pearson score for our system using linear regression scored the highest. Table 2 presents a more thorough examination of results.

	MSRpar	MSRvid	SMT-eur	On-WN	SMT-news	**ALLnrm**	**Mean**	**ALL**
task-baseline	.4334	.2996	**.4542**	.5864	.3908	.6732 (85)	.4356 (70)	.3110 (87)
LReg	.5460	.7818	.3547	.5969	**.4137**	.8043 (36)	.5699 (41)	.6497 (33)
ELReg	.5480	.7844	.3513	.6040	.3607	.8048 (34)	.5654 (44)	**.6622** (27)
ERReg	**.5610**	**.7857**	.3568	**.6214**	.3732	**.8083** (28)	**.5755** (37)	.6573 (28)

Table 1: Pearson's r scores for the official submission. ALLnrm: Pearson correlation after the system outputs for each dataset are fitted to the gold standard using least squares, and corresponding rank. Mean: Weighted mean across the 5 datasets, where the weight depends on the number of pairs in the dataset. ALL: Pearson correlation with the gold standard for the five datasets, and corresponding rank. Parentheses indicate official rank out of 87 systems.

	MSRpar	MSRvid	SMT-eur	On-WN	SMT-news	**Mean**	**ALL**
system-baseline	.5143	.7736	.3574	.5017	.3867	.5343	.6542
+Neu	.5243	.7811	.3772	.4860	.3410	.5318	.6643
+Ew	.5267	.7787	**.3853**	.5237	**.4495**	.5560	.6724
+Sel	.4973	.7684	.3129	.4812	.4016	.5306	.6492
+Neu, +Ew	**.5481**	.7831	.2751	.5576	.3424	.5404	.6647
+Neu, +Sel	.5230	.7775	.3724	.5327	.3787	**.5684**	**.6818**
+Ew, +Sel	.5239	.7728	.2842	.5191	.4038	.5320	.6554
+Neu, +Ew, +Sel	.5441	**.7835**	.2644	**.5877**	.3578	.5472	.6645

Table 2: Pearson's r scores for runs based on various combinations of features. Mean: Weighted mean across the 5 datasets, where the weight depends on the number of pairs in the dataset. ALL: Pearson correlation with the gold standard for the five datasets, and corresponding rank.

Discussion. In the aggregate, we see that each of the similarity metrics has the ability to improve results when used with the right combination of other features. For example, while selector similarity by itself does not seem to help overall, using this metric in conjunction with the neural model of similarity gives us our best results. Interestingly, the opposite is true of eigenword similarity, where the best results are seen when they are independent of selectors or the neural models. The decreased correlations can be accounted for by the new features introducing over fitting, and one should note that no such reductions in performance are significant compared to the baseline, where as our best performance is a significant ($p < 0.05$) improvement.

There are a few potential directions for future improvements. We did not tune our system differently for different data sets although there is evidence of specific features favoring certain data sets. In the case of the neural model of similarity we expect that deriving phrase level representations from the sentences and utilizing the dynamic pooling layer should give us a more thorough measure of similarity beyond the sentence-level vectors we used in this work. For eigenwords, we would like to experiment with context-aware vectors as was described in (Dhillon et. al, 2011). Lastly, we were only able to acquire selectors for nouns, but we believe introducing selectors for other parts of speech will increase the power of the selector similarity metric.

4 Conclusion

In this paper, we described two novel word-level similarity metrics, namely *eigenword similarity* and *selector similarity*, that leverage Web-scale corpora in order to build word-level vector representations. Additionally, we explored the use of a vector-model at the sentence-level by unfolding a neural model of semantics. We utilized these metrics in addition to knowledge-based similarity, and surface-level similarity metrics in a regression system to estimate similarity at the sentence level. The performance of the features varies significantly across corpora but at the aggregate, eigenword similarity, selector similarity, and the neural model of similarity all are shown to be capable of improving performance beyond standard surface-level and WordNet similarity metrics alone.

References

Eneko Agirre, Daniel Cer, Mona Diab and Aitor Gonzalez. 2012. The SemEval-2012 Task-6 : A Pilot on Semantic Textual Similarity. *In Proceedings of the 6th International Workshop on Semantic Evaluation (SemEval 2012)*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing : deep neural networks with multitask learning. *In International Conference on Machine Learning*. Pages 160-167.

Paramveer Dhillon, Dean Foster and Lyle Ungar. 2011. Multiview learning of word embeddings via CCA. *In Proceedings of Neural Information Processing Systems*.

Jay Jiang and David Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. *In Proceedings on International Conference on Research in Computational Linguistics*, pages 1933.

Dekang Lin. 1997. Using syntactic dependency as local context to resolve word sense ambiguity. *In Proceedings of the 35th annual meeting of Association for Computational Linguistics*, pages 64-71.

Ted Pedersen, Siddharth Patwardhan and Jason Michelizzi. 2004. WordNet::Similarity-measuring the relatedness of concepts. *In Proceedings of the North American Chapter of the Association for Computational Linguistics*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, G. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. Vol 12.2825-2830

Hansen A. Schwartz and Fernando Gomez. 2008. Acquiring knowledge from the web to be used as selectors for noun sense disambiguation. *In Proceedings of the Twelfth Conference on Computational Natural Language Learning*.

Hansen A. Schwartz and Fernando Gomez. 2011. Evaluating semantic metrics on tasks of concept similarity. *In Proceedings of the twenty-fourth Florida Artificial Intelligence Research Society*. Palm Beach, Florida: AAAI Press.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng and Christopher Manning. 2011. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. *In Advances in Neural Information Processing Systems*.

Mark Stevenson and Mark A. Greenwood. 2005. A Semantic Approach to IE Pattern Induction. *In Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 379386.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. *In Proceedings of the annual meeting of Association for Computational Linguistics*.