SRIUBC-Core: Multiword Soft Similarity Models for Textual Similarity

Eric Yeh
SRI International
Menlo Park, CA USA
yeh@ai.sri.com

Eneko Agirre
University of Basque Country
Donostia, Basque Country
e.agirre@ehu.es

Abstract

In this year's Semantic Textual Similarity evaluation, we explore the contribution of models that provide soft similarity scores across spans of multiple words, over the previous year's system. To this end, we explored the use of neural probabilistic language models and a TF-IDF weighted variant of Explicit Semantic Analysis. The neural language model systems used vector representations of individual words, where these vectors were derived by training them against the context of words encountered, and thus reflect the distributional characteristics of their usage. To generate a similarity score between spans, we experimented with using tiled vectors and Restricted Boltzmann Machines to identify similar encodings. We find that these soft similarity methods generally outperformed our previous year's systems, albeit they did not perform as well in the overall rankings. A simple analysis of the soft similarity resources over two word phrases is provided, and future areas of improvement are described.

1 Introduction

For this year's Semantic Textual Similarity (STS) evaluation, we built upon the best performing system we deployed last year with several methods for exploring the soft similarity between windows of words, instead of relying just on single token-to-token similarities. From the previous year's evaluation, we were impressed by the performance of features derived from bigrams and skip bigrams. Bigrams capture the relationship between two concurrent words, while skip bigrams can capture longer distance relationships. We found that characterizing the overlap in skip bigrams between the sentences in a STS problem pair proved to be a major contributor to last year's system's performance.

Skip bigrams were matched on two criteria, lexical matches, and via part of speech (POS). Lexical matching is brittle, and even if the match were made on lemmas, we lose the ability to match against synonyms. We could rely on the token-to-token similarity methods to account for these non-lexical similarities, but these do not account for sequence nor dependencies in the sentencees. Using POS based matching allows for a level of generalization, but at a much broader level. What we would like to have is a model that can capture these long distance relationships at a level that is less broad than POS matching, but allows for a soft similarity scoring between words. In addition, the ability to encompass a larger window without having to manually insert skips would be desirable as well.

To this end we decided to explore the use of neural probabilistic language models (NLPM) for capturing this kind of behavior (Bengio et al., 2003). NLPMs represent individual words as real valued vectors, often at a much lower dimensionality than the original vocabulary. By training these representations to maximize a criterion such as log-likelihood of target word given the other words in its neighborhood, the word vectors themselves can capture commonalities between words that have been used in similar contexts. In previous studies, these vectors themselves can capture distributionally derived similarities, by directly comparing the word vectors themselves using simple measures such as

*Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task*, pages 155–161, Atlanta, Georgia, June 13-14, 2013. ©2013 Association for Computational Linguistics

Euclidean distance (Collobert and Weston, 2008).

In addition, we fielded a variant of Explicit Semantic Analysis (Gabrilovich and Markovitch, 2009) that used TF-IDF weightings, instead of using the raw concept vectors themselves. From previous experiments, we found that using TF-IDF weightings on the words in a pair gave a boost in performance over sentence length comparisons and above, so this simple modification was incorporated into our system.

In order to identify the contribution of these soft similarity methods against last year's system, we fielded three systems:

1. **System 1**, the system from the previous year, incorporating semantic similarity resources, precision focused and Bilingual Evaluation Understudy (BLEU) overlaps (Papineni et al., 2002), and several types of skip-bigrams.

2. **System 2**, features just the new NLPM scores and TFIDF-ESA.

3. **System 3**, combines System 1 and System 2.

For the rest of this system description, we briefly describe the previous year's system (System 1), the TFIDF weighted Explicit Semantic Analysis, and the NLPM systems. We then describe the experiment setup, and follow up with results and analysis.

2 System 1

The system we used in SemEval 2012 consisted of the following components:

1. Resource based word-to-word similarities, combined using a Semantic Matrix (Fernando and Stevenson, 2008).

2. Cosine-based lexical overlap measure.

3. Bilingual Evaluation Understudy (BLEU) (Papineni et al., 2002) lexical overlap.

4. Precision focused part-of-speech (POS) features.

5. Lexical match skip-bigram overlap.

6. Precision focused skip-bigram POS features.

The Semantic Matrix assesses similarity between a pair s_1 and s_2 by summing over all of the word to word similarities between the pair, subject to normalization, as given by Formula 1.

$$\mathrm{sim}(s_1, s_2) = \frac{\mathbf{v}_1^T \mathbf{W} \mathbf{v}_2}{\|\mathbf{v_1}\| \, \|\mathbf{v_2}\|} \tag{1}$$

The matrix $\mathbf{W}$ is a symmetric matrix that encodes the word to word similarities, derived from the underlying resources this is drawn from. From the previous year's assessment, we used similarities derived from Personalized PageRank (Agirre et al., 2010) over WordNet (Fellbaum, 1998), the Explicit Semantic Analysis (Gabrilovich and Markovitch, 2009) concept vector signatures for each lemma, and the Dekang Lin Proximity-based Thesaurus [1].

The cosine-based lexical overlap measure simply measures the cosine similarity, using strict lexical overlap, between the sentence pairs. The BLEU, precision focused POS, and skip-bigrams are directional measures, which measure how well a target sentence matches a source sentence. To score pair of sentences, we simply averaged the score where one sentence is the source, the other the target, and then vice versa. These directional measures were originally used as a precision focused means to assess the quality of machine translations output against reference translations. Following (Finch et al., 2005), these measures have also been shown to be good for assessing semantic similarity between pairs of sentences.

For BLEU, we measured how well ngrams of order one through four were matched by the target sentence, matching solely on lexical matches, or POS matches. Skip bigrams performed similarly, except the bigrams were not contiguous. The precision focused POS features assess how well each POS tag found in the source sentence has been matched in the target sentence, where the matches are first done via a lemma match.

To combine the scores from these features, we used the LIBSVM Support Vector Regression (SVR) package (Chang and Lin, 2011), trained on the training pair gold scores. Per the previous year, we used a radial basis kernel with a degree of three.

[1] http://webdocs.cs.ualberta.ca/ lindek/downloads.htm

For a more in-depth description of System 1, please refer to (Yeh and Agirre, 2012).

3 TFIDF-ESA

This year instead of using Explicit Semantic Analysis (ESA) to populate a word-by-word similarity matrix, we used ESA to derive a similarity score between the sentences in a STS pair. For a given sentence, we basically treated it as an IR query against the ESA concept-base: we tokenized the words, extracted the ESA concept vectors, and performed a TFIDF weighted average to arrive at the sentence vector. A cutoff of the top 1000 scoring concepts was further applied, per previous experience, to improve performance. The similarity score for two sentence vectors was computed using cosine similarity.

4 Neural Probabilistic Language Models

Neural probabilistic language models represent words as real valued vectors, where these vectors are trained to jointly capture the distributional statistics of their context words and the positions these words occur at. These representations are usually at a much lower dimensionality than that of the original vocabulary, forcing some form of compression to occur in the vocabulary. The intent is to train a model that can account for words that have not been observed in a given context before, but that word vector has enough similarity to another word that has been encountered in that context before.

Earlier models simply learnt how to model the next word in a sequence, where each word in the vocabulary is initially represented by a randomly initialized vector. For each instance, a larger vector is assembled from the concatenation of the vectors of the words observed, and act as inputs into a model. This model itself is optimized to maximize the likelihood of the next word in the observed sequence, with the errors backpropagated through the vectors, with the parameters for the vectors being tied (Bengio et al., 2003).

In later studies, these representations are the product of training a neural network to maximize the margin between the scores it assigns to observed "correct" examples, which should have higher scores, and "corrupted examples," where the token of interest is swapped out to produce an incorrect example and preferably a lower score. As shown in (Collobert and Weston, 2008) and then (Huang et al., 2012), simple distance measures using the representations derived from this process are both useful for assessing word similarity and relatedness. For this study, we used the contextually trained language vectors provided by (Huang et al., 2012), which were trained to maximize the margin between training pairs and to account for document context as well. The dimensionality of these vectors was 50.

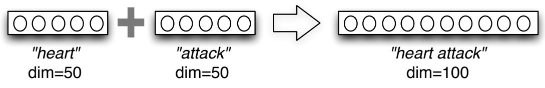

Figure 1: Vector Window encoding for the phrase *"heart attack."*

As we are interested in capturing information at a level greater than individual words, we used two methods to combine these NLPM word vectors to represent an order n ngram: a **Vector Window** where we simply concatenated the word vectors, and one that relied on encodings learnt by **Restricted Boltzmann Machines**.

For this work, we experimented with generating encodings for ngrams sized 2,3,5,10, and 21. The smaller sizes correspond to commonly those commonly used to match ngrams, while the larger ones were used to take advantage of the reduced sparsity. Similarities between a pair of ngram encodings is given similarity of their vector encodings.

4.1 Vector Window

The most direct way to encode an order n ngram as a vector is to concatenate the n NLPM word vectors together, in order. For example, to encode *"heart attack"*, the vectors for *"heart"* and *"attack"*, both with dimensionality 50, are linked together to form a larger vector with dimensionality 100 (Figure 1).

For size n vector windows where the total number of tokens is less than n, we pad the left and right sides of the window with a "negative" token, which was selected to be a vector that, on the average, is anticorrelated with all the vectors in the vocabulary.

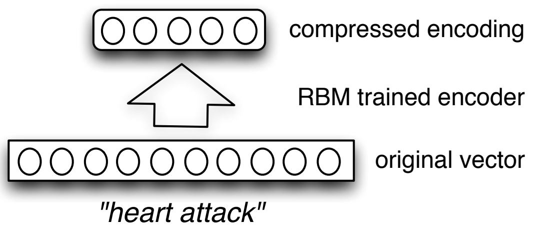

Figure 2: Using a RBM trained compressor to generate a compressed encoding of *"heart attack."*

4.2 Restricted Boltzmann Machines

Although the word vectors we used were trained against a ten word context, the vector windows may not be able to describe similarities at multiword level, as the method is still performing comparisons at a word-to-word level. For example the vector window score for the related phrases *heart attack* and *cardiac arrest* is 0.35. In order to account for similarities at a multiword level, we trained Restricted Boltzmann Machines (RBM) to further encode these vector windows (Hinton, 2002). A RBM is a bipartite undirected graphical model, where the only edges are between a layer of input variables and a layer of latent variables. The latent layer consists of sigmoid units, allowing for non-linear combinations of the inputs. The training objective is to learn a set of weights that maximize the likelihood of training observations, and given the independences inherent, in the model it can be trained quickly and effectively via Contrastive Divergence. The end effect is the system attempts to force the latent layer to learn an encoding of the input variables, usually at a lower dimensionality. In our case, by compressing their distributional representations we hope to amplify significant similarities between multiword expressions, albeit for those of the same size.

To derive a RBM based encoding, we first generate a vector window for the ngram, and then used the trained RBM to arrive at the compressed vector (Figure 2). As before, we derive a similarity score between two RBM based encodings by comparing their cosine distance.

Following the above example, the vectors from an RBM trained system for *heart attack* and *cardiac arrest* score the pair at a higher similarity, 0.54. For phrases that are unrelated, comparing *door key* with *cardiac arrest* gives a score of -0.14 with the vector window, and RBM this is -0.17.

To train a RBM encoder for order n ngrams, we generated n sized vector windows over ngrams drawn from the English language articles in Wikipedia. The language dump was filtered to larger sized articles, in order to avoid pages likely to be content-free, such as redirects. The training set size consisted of 35,581,434 words, which was split apart into 1,519,256 sentences using the OpenNLP sentence splitter tool [2]. The dimensionality of the encoding layer was set to 50 for window sizes 2,3,5, and 200 for the larger windows.

4.3 Combining word and ngram similarity scores

In order to produce an overall similarity score, we used a variant of the weighted variant of the similarity combination method given in (Mihalcea et al., 2006). Here, we generated a directional similarity score from a source to target by the following,

$$\mathrm{sim}(S,T) = \frac{\sum_{s \in S} maxSim(s,T)}{|S|} \tag{2}$$

where $maxSim(s,T)$ represents the maximum similarity between the token s and the set of tokens in the target sentence, T. In the case of ngrams with order 2 or greater, we treat each ngram as a token for the combination.

$$\mathrm{avgsim}(T_1,T_2) = \frac{1}{2}\left(\mathrm{sim}(T_1,T_2) + \mathrm{sim}(T_2,T_1)\right) \tag{3}$$

Unlike the original method, we treated each term equally, in order to account for ngrams with order 2 and above. We also did not filter based off of the part of speech, relying on the scores themselves to help perform the filtering.

In addition to the given word window sizes, we also directly assess the word-to-word similarity scores by comparing the word vectors directly, using a window size of one.

5 Evaluation Setup

System 2, the TFIDF-ESA score for a pair is a feature. For each of the given ngram sizes, we treated

[2]http://opennlp.apache.org/

Training (2012)	Test (2013)
Surprise1 (ONWN)	FNWN
MSRPar	Headlines
Surprise1 (ONWN)	ONWN
Surprise2 (SMT)	SMT

Table 1: Train (2012) and Test (2013) sets used to train the regressors.

the ngram similarity scores from the Vector Window and RBM methods as individual features. System 3 combines the features from System 2 with those from System 1. For Systems 2 and 3, the SVR setup used by System 1 was used to develop scorers. As no training immediate training sets were provided for the evaluation sets, we used the train and test partitions given in Table 1, training on both the 2012 train and test data, where gold scores were available.

6 Results and Discussion

The results of our three runs are given in the top half of Table 2. To get a better sense of the contribution of the new components, we also ran the NLPM vector window and RBM window models and TFIDF-ESA components individually against the test sets. The NLPM system was trained using the same SVR setup as the main experiment.

In order to provide a lexical match comparison for the NLPM system, we experimented with a ngram matching system, where ngrams of size 1,2,3,5,10, and 21 were used to generate similarity scores via the same combination method as the NLPM models. Here, hard matching was performed, where matching ngrams were given a score of 1, else 0. Again, we used the main experiment SVR setup to combine the scores from the various ngram sizes.

We found that overall the previous year's system did not perform adequately on the evaluation datasets, short of the *headlines* dataset. Oddly enough, TFIDF-ESA by itself would have arrived at a good correlation with *OnWN*: one possible explanation for this would be the fact that TFIDF-ESA by itself is essentially an order-free "bag of words" model that assesses soft token to token similarity. As the other systems incorporate either some notion of sequence and/or require strict lexical matching, it is possible that characterization does not help with the *OnWN* sense definitions.

Combining the new features with the previous year's system gave poorer performance; a preliminary assessment over the training sets showed some degree of overfitting, likely due to high correlation between the NLPM features and last year's directional measures.

When using the same combination method, ngram matching via lexical content over ngrams gave poorer results than those from NLPM models, as given in Table 2. This would also argue for identifying better combination methods than the averaged maximum similarity method used here.

What is interesting to note is that the NLPM and TFIDF-ESA systems do not rely on any part of speech information, nor hand-crafted semantic similarity resources. Instead, these methods are derived from large scale corpora, and generally outperformed the previous year's system which relied on that extra information.

To get a better understanding of the NLPM and TFIDF-ESA models, we compared how the components would score the similarity between pairs of two word phrases, given in Table 3. At least over this small sampling we genearted, we found that in general the RBM method tended to have a much wider range of scores than the Vector Window, although both methods were very correlated. Both systems had very low correlation with TFIDF-ESA.

7 Future Work

One area of improvement would be to develop a better method for combining the various ngram similarity scores provided by the NLPMs. When using lexical matching of ngrams, we found that the combination method used here proved inferior to the directional measures from the previous year's systems. This would argue for a better way to use the NLPMs. As training STS pairs are available with gold scores, this would argue for some form of supervised training. For training similarities between multiword expressions, proxy measures for similarity, such as the Normalized Google Distance (Cilibrasi and Vitányi, 2004), may be feasible.

Another avenue would be to allow the NLPM methods to encode arbitrary sized text spans, as the current restriction on spans being the same size is

System	headlines	OnWN	FNWN	SMT	mean	rank
SRIUBC-system1 (Baseline)	0.6083	0.2915	0.2790	0.3065	0.4011	66
SRIUBC-system2 (NLPM, TFIDF-ESA)	0.6359	0.3664	0.2713	0.3476	0.4420	57
SRIUBC-system3 (Combined)	0.5443	0.2843	0.2705	0.3275	0.3842	70
NLPM	0.5791	0.3157	0.3211	0.2698	0.3714	
TFIDF-ESA	0.5739	0.7222	0.1781	0.2980	0.4431	
Lex-only	0.5455	0.3237	0.2095	0.3146	0.3483	

Table 2: Pearson correlation of systems against the test datasets (top). The test set performance for the new Neural Probabilistic Language Model (NLPM) and TFIDF-ESA components are given, along with a lexical-only variant for comparison (bottom).

String 1	String 2	Vec. Window	RBM Window	TFIDF-ESA
heart attack	cardiac arrest	0.354	0.544	0.182
door key	cardiac arrest	-0.14	-0.177	0
baby food	cat food	0.762	0.907	0.079
dog food	cat food	0.886	0.914	0.158
rotten food	baby food	0.482	0.473	0.071
frozen solid	thawed out	0.046	-0.331	0.102
severely burnt	frozen stiff	-0.023	-0.155	0
uphill slog	raced downhill	0.03	-0.322	0.043
small cat	large dog	0.817	0.905	0.007
ran along	sprinted by	0.31	0.238	0.004
ran quickly	jogged rapidly	0.349	0.327	0.001
deathly ill	very sick	0.002	0.177	0.004
ran to	raced to	0.815	0.829	0.013
free drinks	drinks free	0.001	0.042	1
door key	combination lock	0.098	0.093	0.104
frog blast	vent core	0.003	0.268	0.004

Table 3: Cosine similarity of two input strings, as given by the vectors generated from the Vector Window size 2, RBM Window size 2, and TFIDF-ESA.

unrealistic. One possibility is to use recurrent neural network techniques to generate this type of encoding.

Finally, the size of the Wikipedia dump used to train the Restricted Boltzmann Machines could be at issue, as 35 million words could be considered small compared to the full range of expressions we would wish to capture, especially for the larger window spans. A larger training corpus may be needed to fully see the benefit from RBMs.

Acknowledgments

Supported by the Artificial Intelligence Center at SRI International. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Artificial Intelligence Center, or SRI International.

References

Eneko Agirre, Montse Cuadros, German Rigau, and Aitor Soroa. 2010. Exploring knowledge bases for similarity. In *Proceedings of the International Conference on Language Resources and Evaluation 2010*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transac-*

tions on Intelligent Systems and Technology, 2:27:1–27:27.

Rudi Cilibrasi and Paul M. B. Vitányi. 2004. The google similarity distance. *CoRR*, abs/cs/0412098.

R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International Conference on Machine Learning, ICML.*

Christine Fellbaum. 1998. *WordNet - An Electronic Lexical Database*. MIT Press.

Samuel Fernando and Mark Stevenson. 2008. A semantic similarity approach to paraphrase detection. In *Computational Linguistics UK (CLUK 2008) 11th Annual Research Colloqium*.

Andrew Finch, Young-Sook Hwang, and Eiichio Sumita. 2005. Using machine translation evaluation techniques to determine sentence-level semantic equivalence. In *Proceedings of the Third International Workshop on Paraphrasing (IWP 2005)*, pages 17–24, Jeju Island, South Korea.

Evgeniy Gabrilovich and Shaul Markovitch. 2009. Wikipedia-based semantic interpretation. *Journal of Artificial Intelligence Research*, 34:443–498.

Geoffrey E. Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. In *Annual Meeting of the Association for Computational Linguistics (ACL).*

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the American Association for Artificial Intelligence (AAAI 2006)*, Boston, Massachusetts, July.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Eric Yeh and Eneko Agirre. 2012. Sri and ubc: Simple similarity features for semantic textual similarity. In *Proceedings of SemEval 2012*.