DCU: Using Distributional Semantics and Domain Adaptation for the Semantic Textual Similarity SemEval-2015 Task 2

Piyush Arora, Chris Hokamp, Jennifer Foster, Gareth J.F.Jones
ADAPT Centre
School of Computing
Dublin City University
Dublin, Ireland
{parora,chokamp,jfoster,gjones}@computing.dcu.ie

Abstract

We describe the work carried out by the DCU team on the Semantic Textual Similarity task at SemEval-2015. We learn a regression model to predict a semantic similarity score between a sentence pair. Our system exploits distributional semantics in combination with tried-and-tested features from previous tasks in order to compute sentence similarity. Our team submitted 3 runs for each of the five English test sets. For two of the test sets, *belief* and *headlines*, our best system ranked second and fourth out of the 73 submitted systems. Our best submission averaged over all test sets ranked 26 out of the 73 systems.

1 Introduction

This paper describes DCU's participation in the SemEval 2015 English Semantic Textual Similarity (STS) task, whose goal is to predict how similar in meaning two sentences are (Agirre et al., 2014). The semantic similarity between two sentences is defined on a scale from 0 (no relation) to 5 (semantic equivalence). Thus, given a sentence pair, our aim is to learn a model which outputs a score between 0 and 5 reflecting the semantic similarity between the two sentences.

We explore distributional representations of words computed using neural networks – specifically Word2Vec vectors (Mikolov et al., 2013) – and we design features which attempt to encode semantic similarity at the sentence level. We also experiment with several methods of data selection, both for training word embeddings, and for selecting training data for our regression models. We submitted three runs for this task: for all three runs, the features used are identical, and the only difference between them is the training instance selection method used.

2 Data and Resources

The training data for the task is comprised of all the corpora from previous years STS tasks: STS-2012, STS-2013 and STS-2014 (Agirre et al., 2012; Agirre et al., 2013; Agirre et al., 2014). The test data is taken from five domains: *answers-forums*, *answers-students*, *belief*, *headlines* and *images*. Two domains (*headlines* and *images*) have some training data available from the previous STS tasks[1] – the other three have been introduced for the first time.

We use the Word2Vec (W2V) representation for computing semantic similarity between two words. We then expand to incorporate the similarity between two sentences. Using W2V, a word can be represented as a vector of D dimensions, with each dimension capturing some aspect of the word's meaning in the form of different concepts learnt from the trained model. We use the `gensim` W2V implementation (Řehůřek and Sojka, 2010).

We use the *text8* Wikipedia corpus to train our general W2V model. This corpus is comprised of 100MB of compressed Wikipedia data.[2] We use the *UMBC* corpus (Han et al., 2013) for building domain-specific W2V models.

[1]http://ixa2.si.ehu.es/stswiki/index.php/Main_Page
[2]http://mattmahoney.net/dc/textdata.html

3 Methodology

3.1 Pre-processing

We perform minimal pre-processing, replacing all hyphens and apostrophes with spaces, and removing all non-alphanumeric symbols from the data. Our general domain model uses the NLTK[3] stop word list for stop word removal and the Porter stemming algorithm (Porter, 1980). Word2Vec handles the stem variations to some extent when it learns the vector representation from the raw input data. Thus for the domain-specific models, we only remove stopwords and do not stem.

3.2 Feature Design

To predict a semantic similarity score, we learn a regression model using the M5P algorithm.[4] We represent a sentence pair using the features described in the folowing subsections.

3.2.1 Cosine Similarity

We have two features representing the cosine similarity between two sentences, $s1$ and $s2$, where the sentences are represented as binary vectors with each dimension indicating the presence of a word. The first feature is the basic cosine similarity between the two sentence vectors and the second is the weighted cosine similarity between the two vectors, where each word is weighted by its inverse collection frequency (ICF).[5]

3.2.2 Word2Vec

Sum W2V: For a given sentence we represent each word by its W2V representation and then sum each word vector in a sentence to find the centroid of the word vectors representing the entire sentence. The cosine of the centroids of the two sentences indicates the similarity between them. Using the sum approach, two features, *sum* and *sum_icf*, are calculated, one corresponding to the basic cosine similarity between the vectors, and the other representing the weighted cosine similarity where, before calculating the centroid, each word vector is multiplied by its ICF weight.

$$sim(\mathbf{s1},\mathbf{s2}) = \frac{\frac{\sum_{i=1}^{n}\vec{s1}_i}{n}.\frac{\sum_{j=1}^{m}\vec{s2}_j}{m}}{\sqrt{(\frac{\sum_{i=1}^{n}\vec{s1}_i}{n})^2}\sqrt{(\frac{\sum_{j=1}^{m}\vec{s2}_j}{m})^2}} \quad (1)$$

Product W2V: Given $s1$ and $s2$, we take the element-wise product of each word vector in $s1$ and $s2$ and store the maximum product value for each word in $s1$ and similarly for $s2$. The Product W2V feature is the average of the maximum weights between each word of $s1$ with $s2$ and vice versa:

$$sim(\mathbf{s1},\mathbf{s2}) = \frac{\sum_{i=1}^{n}(max\sum_{j=1}^{m}\frac{(\vec{s1}_i.\vec{s2}_j)}{\sqrt{(\vec{s1}_i)^2}\sqrt{(\vec{s2}_j)^2}})}{n} + \frac{\sum_{j=1}^{m}(max\sum_{i=1}^{n}\frac{(\vec{s1}_i.\vec{s2}_j)}{\sqrt{(\vec{s1}_i)^2}\sqrt{(\vec{s2}_j)^2}})}{m} \quad (2)$$

The sum and product W2V models are inspired by the composition models of Mitchell and Lapata (2008) and semantic similarity measures of Mihalcea et al. (2006).

Domain-specific Cosine Similarity: Good coverage is obtained using the *text8* corpus to train the W2V model. However, we also want to explore the performance with respect to an *in-domain* W2V model. So, for each of the test corpora, we first extract a corpus of similar sentences from the *UMBC* corpus by selecting up to 500 sentences for each content word in the test corpus and then use the extracted dataset to train a W2V model that has better coverage of the test domain. Using the domain-specific W2V corpus, we compute the feature *domain_w2v_cosine_similarity* in a similar fashion to the Sum W2V feature – we compute the centroid vector of the content words in each sentence and then compute the cosine between the two centroids.

Syntax: We also hypothesize that two semantically similar sentences should have high overlap between their nouns, verbs, adjectives and adverbs. For each coarse-grained POS tag (`NN*`, `VB*`, `JJ*` and `RB*`) we calculate the W2V cosine similarity between all words from $s1$ and $s2$ which have the same POS tag (using the Sum W2V combination method). For each coarse-grained POS tag, we also calculate the number of lexical matches with that particular POS tag.

[3] `http://www.nltk.org/`

[4] We used the weka implementation: `http://www.cs.waikato.ac.nz/ml/weka/` without performing any extra hyper-parameter optimization.

[5] ICF is calculated using word frequency from the *wikipedia* 2011 dump.

We also parse each sentence using the Stanford parser (Manning et al., 2014) and look for dependency relation overlap between $s1$ and $s2$.[6] We concentrate on six dependency relations – `nsubj`, `dobj`, `det`, `prep`, `amod` and `aux`. For each relation we calculate the degree of overlap between the occurrences of this relation in the two sentences. We have two notions of relation overlap: a non-lexicalized version which just counts the relation itself (e.g. `nsubj`) and a lexicalized version which counts the relation and the two tokens it connects (e.g. `nsubj_word1_word2`).

3.2.3 Monolingual Alignment

We compute the monolingual alignment between the two sentences using the word aligner introduced in (Sultan et al., 2014). Their system aligns related words in a sentence pair by exploiting semantic and contextual similarities of the words. From the aligned sentences, we then extract two features: *percent_aligned_source* and *percent_aligned_target*, which represent the fraction of tokens in each sentence which have an alignment in the other sentence. The intuition behind these features is that sentences which are semantically similar should have a higher fraction of aligned tokens, since alignments constitute either identical strings or paraphrases.

3.2.4 TakeLab

The Takelab system (Šarić et al., 2012) was the top performing system in STS-2012 task. Their system used support vector regression models with multiple features measuring word overlap similarity and syntactic similarity. We find that adding the Takelab features provides additional knowledge to our system and improves performance for the training datasets. We add the 21 features of the Takelab system to our feature set.

3.3 Training instance selection

After designing features to model semantic similarity between two sentences, the next important task is to select the training corpus for learning the weights for these features. Out of the five test sets for STS-2015, we only have in-domain training corpora for the *headlines* and *images* data sets. We hypothesize that finding vocabulary similarity between the entire training and test corpus could be used to select more similar corpus for training of the system. We calculate the similarity between each of the corpora we have from previous STS tasks and each of the test corpora. Using the entire corpus vocabulary as a vector we find the cosine similarity between different corpora using the TFIDF (Manning et al., 2008), LSI (Hofmann, 1999), LDA (Blei et al., 2003b) and HLDA (Blei et al., 2003a) measures.

Next, we describe the mechanism we used for training data selection for each run:

1. *Run-1*: For the two corpora for which we have prior training data we took the previous years' training data. For the other test corpora we select the most similar corpus from the previous years' training data based on the corpus vector cosine similarity, where each word in a vector is replaced by its TFIDF weight. The training corpora we selected are as follows:
   Images: Images_2014, *Headlines*: Headlines-_2014, *Belief*: Deft_Forum_2014, *Answers-students*: MSRVid_2012_train, *Answers-forums*: Deft_Forum_2014

2. *Run-2*: We want to make sure that the training data has instances similar to the test samples. To capture diversity in our training corpus we compute corpus vector cosine similarity where each word is replaced by its TFIDF weight, then we merge the top three most similar training corpora for each test set as shown below:
   Images: Images_2014 + MSRVid_2012_train + MSRVid_2012_test
   Headlines: Headlines_2013 + Headlines_2014 + MSRPar_2012_train
   Belief: Deft_Forum_2014 + Headlines_2014 + Headlines_2013
   Answers-students: OnWn_2012_test + MSR-Vid_2012_train + MSRPar_2012_test
   Answers-forums: Deft_forum_2014 + SMT-_2012_train + MSRPar_2012_train

   For each test set instance, we find the five[7] most similar instances from the merged training

[6]Parsing is carried out on the raw sentences.

[7]Five was empirically chosen by experimenting with different values on the training data.

Test Set	**Baseline**	**Run-1**	**Run-2**	**Run-3**	**Top System**	Our Rank
Images	0.6039	0.8394	0.835	**0.8434**	0.8713	19
Headlines	0.5312	**0.8284**	0.8187	0.8181	0.8417	4
Belief	0.6517	0.5464	**0.7549**	0.6977	0.7717	2
Answers-students	0.6647	**0.6582**	0.6233	0.6108	0.7879	47
Answers-forum	0.4453	0.5556	0.5628	**0.653**	0.739	30
Mean		0.7192	0.734	**0.7369**		26

Table 1: Results of our final runs compared to the baseline and the best system for each test set.

corpus (similar instances are computed using cosine similarity between the feature vectors). By combining these five training instances for all test instances and removing duplicates, we form a more focused training set which is expected to capture the test set diversity more effectively.

3. *Run-3*: In this variant, we do not want to limit ourselves to just the top three corpora, so we merge all the training data and then look for the five most similar training instances for each test instance to form a focused training set.

4 Results

Table 1 shows the results of our systems on the five test sets. For the test sets *answers-forum* and *belief* there was a considerable difference in the results across the three runs, indicating that selecting training instances has a significant effect on performance. For these two datasets across two runs the absolute difference in the Pearson coefficient is about 10% for *answers-forum* and about 20% for the *belief* dataset. Overall, our best system rank is 26 out of 73. If we look at the results for individual test sets, it seems our approach works well for the *belief*, *headlines* and *image* test set but performs poorly for the *answer-student* and *answer-forum* test sets. For the *belief* test set our Run-2 was ranked 2nd overall and for the *headlines* test set our Run-1 was ranked 4th overall. For the *images* test set, the results are competitive – the absolute difference in the Pearson value between our best run and the best system is only 0.03. Thus, apart from two corpora, *answers-students* and *answer-forums*, our approach performed quite well.

We analyzed the features using GradientBoostingRegressor[8] for all the training sets. The feature importance varies slightly across different domains. For all datasets, we remove features with gini importance[9] < 0.01, then we look for the features which are present in at least three of the different domain for this year's test set. The features that performed well are shown in Table 2.

Our Features
sum_icf, sum, product, domain_w2v_cosine, percent_aligned_target, percent_source_target, nn_w2v, vb_w2v, jj_w2v, nsub_1, cosine, cosine_icf
TakeLab Features
wn_sim_match, weighted_word_match, weighted_word_match, dist_sim, weighted_dist_sim, weighted_dist_sim, relative_len_difference, relative_ic_difference

Table 2: Important features.

5 Conclusions

All of our runs have the same features, but use different training corpora to learn the weights. We thus show that training data selection can have an impact on the performance of a model, especially for a novel genre. Using Word2Vec to find semantic similarity between a sentence pair proved to be effective. Furthermore, composing W2V features in different ways can help to reveal new information about semantic similarity.

Investigating the test sets where we failed to perform well, *answer-forums* and *answer-students*, reveals that we need to handle phrasal information more effectively by, for example, handling negation,

[8] `http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html`

[9] `http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm`

devising measures to compare the sentences at the entity level and making better use of parser output.

Acknowledgments

We would like to thank Rasoul Kaljahi for helping with running some of the preliminary experiments using tree kernels and Debasis Ganguly for his suggestions and guidance. This research is supported by Science Foundation Ireland (SFI) as a part of the CNGL Centre for Global Intelligent Content at DCU (Grant No: 12/CE/I2267).

References

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 385–393, Montréal, Canada.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic Textual Similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. SemEval-2014 Task 10: Multilingual Semantic Textual Similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 81–91, Dublin, Ireland.

David M. Blei, Thomas L. Griffiths, Michael I. Jordan, and Joshua B. Tenenbaum. 2003a. Hierarchical Topic Models and the Nested Chinese Restaurant Process. In *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003]*, pages 17–24, Vancouver and Whistler, British Columbia, Canada.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003b. Latent Dirichlet Allocation. *Journal of Machine Learning Research - Volume 3*, pages 993–1022.

Lushan Han, Abhay L. Kashyap, Tim Finin, James Mayfield, and Johnathan Weese. 2013. UMBC_EBIQUITY-CORE: Semantic Textual Similarity Systems. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*, pages 42–54, Atlanta, Georgia, USA.

Thomas Hofmann. 1999. Probabilistic Latent Semantic Indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57, Berkeley, California, USA.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. Scoring, term weighting and the vector space model. *Introduction to Information Retrieval*, pages 118–120.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*, pages 775–780, Boston, Massachusetts.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013. Exploiting Similarities among Languages for Machine Translation. *CoRR*, abs/1309.4168.

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

Martin F Porter. 1980. An algorithm for suffix stripping. *Program - Volume 14*, pages 130–137.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta.

Arafat Md Sultan, Steven Bethard, and Tamara Sumner. 2014. Back to Basics for Monolingual Alignment: Exploiting Word Similarity and Contextual Evidence. *Transactions of the Association of Computational Linguistics – Volume 2, Issue 1*, pages 219–230.

Frane Šarić, Goran Glavaš, Mladen Karan, Jan Šnajder, and Bojana Dalbelo Bašić. 2012. TakeLab: Systems for Measuring Semantic Text Similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, SemEval '12, pages 441–448, Montréal, Canada.