Azmat: Sentence Similarity using Associative Matrices

Evan Jaffe Lifeng Jin David King Marten van Schijndel
Department of Linguistics
The Ohio State University
{jaffe.59, jin.544, king.2138}@osu.edu, vanschm@ling.osu.edu

Abstract

This work uses recursive autoencoders (Socher et al., 2011), word embeddings (Pennington et al., 2014), associative matrices (Schuler, 2014) and lexical overlap features to model human judgments of sentential similarity on SemEval-2015 Task 2: English STS (Agirre et al., 2015). Results show a modest positive correlation between system predictions and human similarity scores, ranking 69th out of 74 submitted systems.

1 Introduction

This work uses a support vector machine (SVM) to determine the similarity of sentence pairs, taking as input the similarity judgments of four subsystems: a set of surface features, unfolding recursive autoencoders (URAE; Socher et al., 2011), Global Vector word embeddings (GloVe; Pennington et al., 2014), and the Schuler (2014) associative matrix approach using the Nguyen et al. (2012) Generalized Categorial Grammar (GCG). Evaluation is run on SemEval 2015 task 2, Semantic Textual Similarity (STS), which includes a corpus of human similarity judgments. The test set consists of 3000 randomly chosen sentence pairs from a corpus of 8500 pairs, which spans five domains (news headlines, image captions, student answers, forum responses, and sentences about belief). Similarity scores range from 0 (no similarity) to 5 (complete semantic equivalence).

2 System Overview

All subsystems in Azmat are trained with sentences from previous SemEval tasks 2012 - 2014 (Agirre et al., 2012; Agirre et al., 2013; Agirre et al., 2014). In total, 15,406 sentences were selected from the Microsoft video, news headlines, images, and paraphrase datasets. The main purpose of the subsystems (excluding surface features) is to generate binarized phrase-structure trees, which are used to create cosine similarity features between multiple levels of paired sentences. The URAE subsystem preprocesses training sentences by parsing them with the Stanford Parser (Klein and Manning, 2003) and then binarizing. The associative matrix and GloVe subsystems use GCG parses of the training sentences, obtained by training the Berkeley parser (Petrov and Klein, 2007) with the Nguyen et al. (2012) GCG reannotated Penn Treebank. GCG parse trees are converted into typed dependency graphs and binarized. Around 2% of the sentences fail to parse; these are omitted from the training set.

2.1 Subsystem Combination

Because vector composition methods vary across subsystems, this work incorporates multiple subsystems to give insight on which composition methods perform better at finding semantic textual similarity. For each sentence, each subsystem generates a single binarized phrase-structure tree with a single embedding labeled at each node. Cosine similarity scores are calculated between each node in one tree and each node in the other tree, allowing comparison between input sentences at and across leaf, phrasal and sentential levels. These similarity scores are used to generate a feature vector for training an SVM regressor with a linear kernel.[1]

[1] http://scikit-learn.org/stable/modules/svm.html

In order to generalize findings across sentence pairs with varying lengths and tree structures, however, similarity scores must be consistently ordered for the SVM and must generate a feature vector of consistent length. To accomodate these constraints, each output node (n) in a tree is assigned a *composition depth* (d_n) based on the depth of its child nodes (a and b):

$$d_n = \begin{cases} 0 & \text{if } n \text{ is a leaf} \\ \max(d_a, d_b) + 1 & \text{otherwise} \end{cases} \tag{1}$$

Similarity between two nodes are grouped with similarities of similar depth (x and y) into a vector (v_{xy}), which is sorted before being concatenated with other depth similarities[2] to form the actual feature vector which will be input to the SVM:

$$|\underbrace{0.8\ 0.7\ 0.3\ldots}_{(d{=}0\ d{=}0)}|\underbrace{0.9\ 0.4\ 0.2\ldots}_{(d{=}0\ d{=}1)}|\cdots \tag{2}$$

The actual ordering of the concatenated depth groups within the vector does not matter to the downstream SVM classifier so long as the ordering is consistent. Each v_{xy} is given a constant length to losslessly capture the similarity of balanced trees up to 50 words in length:[3]

$$|v_{xy}| = \frac{50}{2^{d_x}} \cdot \frac{50}{2^{d_y}} \tag{3}$$

Each depth-pair subvector is duplicated up to the needed length before being re-sorted. This approach is analogous to a lossless version of the dynamic pooling used by Socher et al. (2011).

Using the above approach, each subsystem generates its own version of the vector in (2). Then each of those vectors is concatenated together to form the entire SVM input vector.

2.2 Surface Features

Surface features include *n*-gram overlap measures of precision, recall, and F-score, where precision and recall are defined as overlap from sentence A to sentence B, and from sentence B to sentence A, respectively. 1- through 3-grams are measured using stemmed[4] and unstemmed lexical items for each of the 3 overlaps, resulting in a total of 18 surface features. These features are based on those used by Das and Smith (2009) for paraphrase detection.

2.3 Unfolding Recursive Autoencoders

Socher et al. (2011) show good results for paraphrase detection by using recursive autoencoders (RAEs) to compose word embeddings into phrasal and sentential embeddings, allowing similarity metrics at various structural levels. Their method uses word embeddings from Turian et al. (2010) as input, along with a binarized phrase-structure parse from the Stanford Parser (Klein and Manning, 2003). Given a binarized parse tree and leaf node embeddings, weight matrices are learned to both encode and decode nodes above the leaves by minimizing reconstruction error. ‘Unfolding’ refers to a learning objective that reconstructs the entire subtree below each node, not just the immediate children. Once a model is trained, the learned encoding matrix can generate embeddings at each node for novel sentences. The current work uses the pre-trained model and code from Socher et al. (2011) to generate features from the previous SemEval task sentences.

2.4 Associative Matrices

The associative matrix subsystem (AM) is inspired by a cognitively-grounded parsing model that stores associations between words as dependency relations (Nguyen et al., 2012; Wu and Schuler, 2011). Dependency-like associations are learned from typed dependency graphs generated from gold Nguyen et al. (2012) GCG annotations of Simple Wikipedia. Dependency-based skip-grams are used to build a co-occurrence matrix for all words, and single value decomposition (SVD; Landauer and Dumais, 1997) generates word embeddings with reduced dimensionality.

Each labeled dependency in the training data is recorded in associative matrices by adding the outer product of the governor and the dependent to the matrix corresponding to the dependency label, creating

[2] Remember that similarities are computed between all nodes in one tree and all nodes in the other tree, which results in some similarities being computed between nodes of different depths.

[3] Consistent lengths permit each v_{xy} to be at a consistent position within the overall feature vector.

[4] NLTK Lancaster Stemmer (Bird et al., 2009; Paice, 1990)

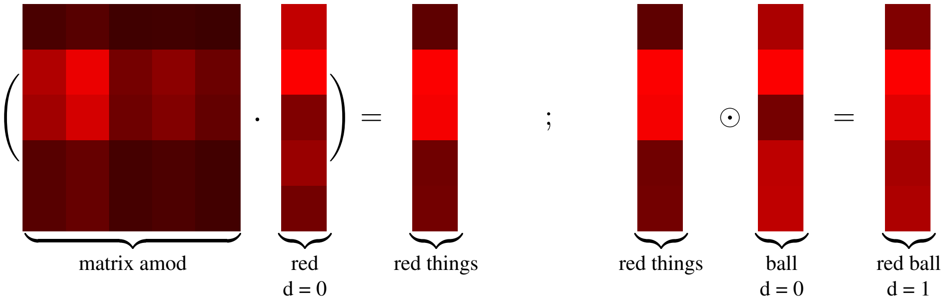

Figure 1: Example vector composition using learned associative matrices. The dependency triple (*red*, *ball*, *amod*) can be composed by first cueing *red* off of the *amod* matrix. The resulting target vector represents a superposition of all governors *red* stands in an *amod* relation to. The target is then pointwise multiplied with the embedding for *ball* to get a final phrasal representation. Note that words are depth 0, and the composition results in an embedding at depth 1.

an associative matrix for each dependency type:

$$M_{deplabel} = \sum_{D} (\bar{u} \otimes \bar{v}) \qquad (4)$$

where (*u, v, deplabel*) is a labeled dependency.

To compose a phrasal embedding, the dependent word embedding is first inner multiplied with the association matrix for the dependency type, a process called cueing, which returns a target vector. Cueing converts the dependent word embedding into the space of its governor, essentially representing the superposed vectors of all governors that the dependent co-occurs with. Finally, the target is pointwise multiplied with the governor embedding, reinforcing the influence of the observed governor and specifying the meaning of the phrase as a combination of the meaning of the dependent and of its governor. See Table 1 for an example. All unknown (OOV) word vectors are filled with ones to avoid contaminating products during composition. As with all subsystems, a single binarized parse tree with an embedding at each node is the result.

2.5 Global Vectors

Due to the success of word embeddings in word similarity judgment tasks (Mikolov et al., 2013), this work also makes use of Global Vector word embeddings (GloVe; Pennington et al., 2014). 300-dimensional GloVe embeddings are trained on 42 billion lower-cased tokens from the Stanford tokenized Common Crawl. These word embeddings are combined using the same GCG structure as the AM subsystem. Each node in the GCG tree is assigned the embedding of that subtree's head word, so the 'red ball' node is assigned the embedding for 'ball'. All OOV word vectors are drawn from a uniform distribution between 0 and 1.

Model	Unk ρ	Known ρ	Test ρ
SUGA	0.5370	0.6118	0.4512
UGA	0.4620	0.5493	
SUA	0.5547	0.6233	
SGA	0.5650	0.6299	
SUG	0.5897	0.6566	

Table 1: Model correlation with human judgments on unknown and known domains in development as each subsystem is omitted (included subsystems are noted: **S** for surface, **U** for URAE, **G** for GloVe, and **A** for AM). Final system performance on test data for the task is also shown at right.

3 Experiments and Error Analysis

For development, 1000 pairs are held out of the training data in jack-knifed batches. Table 1 shows how the system performs when each subsystem is omitted. Each model is designated using the first letter of each subsystem, so the full model is named *SUGA*. Table 1 (left) shows the performance of the system when all of the held-out pairs are from a single domain (e.g., news headlines) and thus approximates the system's performance on unknown domains. Table 1 (middle) shows the performance when the held-out pairs are distributed evenly across

Dataset	Leaf	Comp	Cross	Full
Belief	0.5435	0.4966	0.4338	0.3587
Forums	0.4871	0.4114	0.4535	0.2933
Headlines	0.6583	0.6389	0.5826	0.5264
Images	0.6276	0.5587	0.5369	0.5145
Students	0.6399	0.5454	0.5222	0.4293
Mean	0.5913	0.5302	0.5058	0.4244
Wt. Mean	0.6103	0.5493	0.5213	0.4491

Table 2: Correlations with human judgments when only certain similarity relations are used: only word-level similarity (leaf), only compositional non-leaf similarity (comp), only similarity between leaf and non-leaf nodes (cross), and permitting all similarities (full). The weighted mean accounts for the proportion of test cases in each dataset.

all domains and so estimates the system's performance on domains that are familiar. SemEval-2015 Task 2 test results are shown in Table 1 (right).[5]

Omission of the surface features results in a sharp performance decrease, showing they capture complementary information to other features. See *UGA* model as compared to the *SUGA* model in Table 1. Also observable in the table is that excluding any one of the three main subsystems (URAE, GloVe, AM) improves performance, which implies the full system overfits to the training data.[6] Since the composition method differs between all three subsystems, and since URAE even uses a different underlying dependency structure, the overfit likely stems from the fact that all three systems are computing leaf/leaf similarity. Overfitting might be reduced by either only using the leaf/leaf similarity from a single system or by tuning the tolerance of the SVM.[7]

Since the development results suggest that the full system overfits, it may be informative to test how the different parts of the compositional framework behave. To test this, the full SUGA system is retrained with some similarity relations removed (see Table 2). When only leaf/leaf similarities are used during training, the system performs the best. This finding is likely due to the ubiquity of word-level similarity/analogy as a task, for which word embeddings such as GloVe were designed. System performance declines when trained only on similarities between non-leaf nodes, suggesting the compositions are less good at reflecting phrasal- and sentence-level similarity. The system becomes even less accurate when only using similarities between leaf nodes and non-leaf nodes, which were hoped to enable the system to capture similarities between more and less general phrases (e.g., between 'red ball' and 'ball'). This finding is somewhat surprising since URAE is thought to capture these types of similarities.

Although leaf/leaf similarities are useful, overreliance on non-compositional nodes causes problems when comparing pairs with more abstract differences. For example, the system rates the following unrelated pair as very similar despite completely different subject-predicate and modifier compositions:

Zoo worker dies after tiger attack
Teacher dies after attack in New Zealand

Further, while coarse feature selection (e.g., removing all non-leaf features) improves performance, it is not a foregone conclusion that composition features are completely uninformative. For example, comparisons between nodes of similar depths (e.g., 0-1, 4-3) might be more informative than node comparisons of dissimilar depths (e.g., 1-7, 6-2), so future work should determine whether there is an information gradient when comparing compositional nodes. Additionally, the fixed length chosen in this work for each depth-paired subvector guarantees a lossless representation of similarities between balanced trees up to 50 words long, but the similarity vectors involving non-leaf nodes become increasingly lossy as the input trees become less balanced. Therefore, the current system possibly underestimates the informativity of non-leaf features.

4 Conclusion

The current work combined surface lexical features with lexical and phrasal tree node similarity features using URAE, GLoVe, and an associative matrix composition system to model sentential similarity. Since phrasal similarity is likely extremely useful in determining sentence similarity, this work provides insight into the use and combination of multiple phrasal similarity systems.

[5] *SUGA* ranked 69th of 74 systems. For full results, see http://alt.qcri.org/semeval2015/task2/index.php?id=results

[6] One example of overfitting is that the larger *SUGA* model performs worse than the smaller *SUG* model for the same *known* dataset (0.6118<0.6566).

[7] The current work uses an untuned tolerance of 0.001.

Acknowledgements

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1343012. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. We would also like to thank the anonymous reviewers for their helpful suggestions and comments.

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. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Guo WeiWei. 2013. sem-2013 shared task: Semantic textual similarity, including a pilot on typed-similarity. In *In *SEM 2013: The Second Joint Conference on Lexical and Computational Semantics, Association for Computational Linguistics*.

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, August. Association for Computational Linguistics and Dublin City University.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Guo WeiWei, Iigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, Denver, CO, June. Association for Computational Linguistics.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly, Beijing.

Dipanjan Das and Noah A. Smith. 2009. Paraphrase identification as probablistic quasi-synchronous recognition. In *Proc. of ACL-IJCNLP*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan.

T.K. Landauer and S.T. Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781:1–12.

Luan Nguyen, Marten van Schijndel, and William Schuler. 2012. Accurate unbounded dependency recovery using generalized categorial grammars. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING '12)*, pages 2125–2140, Mumbai, India.

Chris D. Paice. 1990. Another stemmer. *SIGIR Forum*, 24:56–61.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of NAACL HLT 2007*, pages 404–411, Rochester, New York, April. Association for Computational Linguistics.

William Schuler. 2014. Sentence processing in a vectorial model of working memory. In *Fifth Annual Workshop on Cognitive Modeling and Computational Linguistics (CMCL 2014)*.

Richard Socher, Eric Huang, Jeffrey Pennington, Andrew Ng, and Christopher Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Neural Information Processing Systems (NIPS)*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proc. of ACL 2010*.

Stephen Wu and William Schuler. 2011. Structured composition of semantic vectors. In *Proceedings of the International Workshop on Semantic Computing*.