Abstract
We describe our system (DT Team) submitted at SemEval-2017 Task 1, Semantic Textual Similarity (STS) challenge for English (Track 5). We developed three different models with various features including similarity scores calculated using word and chunk alignments, word/sentence embeddings, and Gaussian Mixture Model (GMM). The correlation between our system’s output and the human judgments were up to 0.8536, which is more than 10% above baseline, and almost as good as the best performing system which was at 0.8547 correlation (the difference is just about 0.1%). Also, our system produced leading results when evaluated with a separate STS benchmark dataset. The word alignment and sentence embeddings based features were found to be very effective.

2 Preprocessing
The preprocessing step involved tokenization, lemmatization, POS-tagging, name-entity recognition and normalization (e.g. pc, pct, % are normalized to pc). The preprocessing steps were same as our DTSim system (Banjade et al., 2016).

3 Feature Generation
We generated various features including similarity scores generated using different methods. We describe next the word-to-word and sentence-to-sentence similarity methods used in our system.

3.1 Word-to-Word Similarity
We used the word2vec (Mikolov et al., 2013)\(^1\) vectorial word representation, PPDB database (Pavlick et al., 2015)\(^2\), and WordNet (Miller, 1995) to compute similarity between words. Please see DTSim system description (Banjade et al., 2016) for additional details.

3.2 Sentence-to-Sentence Similarity
3.2.1 Word Alignment Method
We lemmatized all content words and aligned them optimally using the Hungarian algorithm (Kuhn, 1955) implemented in the SEMILAR Toolkit (Rus et al., 2013). The process is the same as finding the maximum weight matching in a weighted bi-partite graph. The nodes are words and the weights are the similarity scores between the word pairs computed as described in § 3.1. In order to avoid noisy alignments, we reset the similarity score below 0.5 (empirically set threshold) to 0. The similarity score was computed as the sum of the scores for all aligned word-pairs divided by the total length of the given sentence pair.

\(^1\)http://code.google.com/p/word2vec/
\(^2\)http://www.cis.upenn.edu/ccb/ppdb/
In some cases, we also applied a penalty for unaligned words which we describe in §3.3.

### 3.2.2 Interpretable Similarity Method

We aligned chunks across sentence-pairs and labeled the alignments, such as Equivalent or Specific as described in Maharjan et al. (2016). Then, we computed the interpretable semantic score as in the DTSim system (Banjade et al., 2016).

### 3.2.3 Gaussian Mixture Model Method

Similar to the GMM model we have proposed for assessing open-ended student answers (Maharjan et al., 2017), we represented the sentence pair as a feature vector consisting of feature sets \( \{7, 8, 9, 10, 14\} \) from §3.4 and modeled the semantic equivalence levels \([0, 5]\) as multivariate Gaussian densities of feature vectors. We then used GMM to compute membership weights to each of these semantic levels for a given sentence pair. Finally, the GMM score is computed as:

\[
\text{mem}_{\text{wt}} = w_i N(x|\mu_i, \sum_i), \quad i \in [0, 5]
\]

\[
\text{gmm}_{\text{score}} = \sum_{i=0}^{5} \text{mem}_{\text{wt}} * i
\]

### 3.2.4 Compositional Sentence Vector Method

We used both Deep Structured Semantic Model (DSSM; Huang et al. (2013)) and DSSM with convolutional-pooling (CDSSM; Shen et al. (2014); Gao et al. (2014)) in the Sent2vec tool\(^3\) to generate the continuous vector representations for given texts. We then computed the similarity score as the cosine similarity of their representations.

### 3.2.5 Tuned Sentence Representation Based Method

We first obtained the continuous vector representations \(V_A\) and \(V_B\) for sentence pair \(A\) and \(B\) using the Sent2vec DSSM or CDSSM models or skip-thought model\(^4\) (Zhu et al., 2015; Kiros et al., 2015). Inspired by Tai et al. (2015), we then represented the sentence pairs by the features formed by concatenating element-wise dot product \(V_A \cdot V_B\) and absolute difference \(|V_A - V_B|\). We used these features in our logistic regression model which produces the output \(\hat{p}_\theta\). Then, we predicted the similarity between the texts in the target pair as \(\hat{y} = r^T \hat{p}_\theta\), where \(r^T = \{1, 2, 3, 4, 5\}\) is the ordinal scale of similarity. To enforce that \(\hat{y}\) is close to the gold rating \(y\), we encoded \(y\) as a sparse target distribution \(p\) such that \(y = r^T p\) as:

\[
p_i = \begin{cases} 
\frac{y - \lfloor y \rfloor}{y - 1}, i = \lfloor y \rfloor + 1 \\
\frac{y - y + 1}{y - 1}, i = \lfloor y \rfloor \\
0, & \text{otherwise}
\end{cases}
\]

where \(1 \leq i \leq 5\) and \(\lfloor y \rfloor\) is floor operation. For instance, given \(y = 3.2\), it would give sparse \(p = [0 0.8 0.2 0]\). For building logistic model, we used training data set from our previous DTSim system (Banjade et al., 2016) and used image test data from STS-2014 and STS-2015 as validation data set.

### 3.2.6 Similarity Vector Method

We generated a vocabulary \(V\) of unique words from the given sentence pair \((A, B)\). Then, we generated sentence vectors as in the followings: \(V_A = (w_{1a}, w_{2a}, \ldots, w_{na})\) and \(V_B = (w_{1b}, w_{2b}, \ldots, w_{nb})\), where \(n = |V|\) and \(w_{ia} = 1\), if \(\text{word}_i\) at position \(i\) in \(V\) has a synonym in sentence \(A\). Otherwise, \(w_{ia}\) is the maximum similarity between \(\text{word}_i\) and any of the words in \(A\), computed as: \(w_{ia} = \max_{j=1}^{\mid A \mid} \sim(w_j, \text{word}_i)\). The \(\sim(w_j, \text{word}_i)\) is cosine similarity score computed using the word2vec model. Similarly, we compute \(V_B\) from sentence \(B\).

### 3.2.7 Weighted Resultant Vector Method

We combined word2vec word representations to obtain sentence level representations through vector algebra. We weighted the word vectors corresponding to content words. We generated resultant vector for \(A\) as \(R_A = \sum_{i=1}^{\mid A \mid} \theta_i \ast \text{word}_i\), where the weight \(\theta_i\) for \(\text{word}_i\) was chosen as \(\text{word}_i \in \{\text{noun} = 1.0, \text{verb} = 1.0, \text{adj} = 0.2, \text{adv} = 0.4, \text{others} (\text{e.g. number}) = 1.0\}\). Similarly, we computed resultant vector \(R_B\) for text \(B\). The weights were set empirically from training data. We then computed a similarity score as the cosine of \(R_A\) and \(R_B\). Finally, we penalized the similarity score by the unalignment score (see §3.3).

### 3.3 Penalty

We applied the following two penalization strategies to adjust the sentence-to-sentence similarity score. It should be noted that only certain similarity scores used as features of our regression models were penalized but we did not penalize...
the scores obtained from our final models. Unless specified, similarity scores were not penalized.

3.3.1 Crossing Score
Crossing measures the spread of the distance between the aligned words in a given sentence pair. In most cases, sentence pairs with higher degree of similarity have aligned words in same position or its neighborhood. We define crossing score as:

$$crs = \frac{\sum_{w_i \in A, w_j \in B, \text{aligned}(w_i, w_j)} |i - j|}{\max(|A|, |B|) \times (#\text{alignments})}$$

where \text{aligned}(w_i, w_j) refers to word \(w_i\) at index \(i\) in \(A\) and \(w_j\) at index \(j\) in \(B\) are aligned. Then, the similarity score was reset to 0.3 if \(crs > 0.7\). The threshold 0.7 was empirically set based on evaluations using the training data.

3.3.2 Unalignment Score
We define unalignment score similar to alignment score (see § 3.2.1) but this time the score is calculated using unaligned words in both \(A\) and \(B\) as:

$$unalign\_score = \frac{|A| + |B| - 2\times(#\text{alignments})}{|A| + |B|}$$

Then, the similarity score was penalized as in the followings:

$$score^* = (1 - 0.4 \times unalign\_score) \times score$$

where the weight 0.4 was empirically chosen.

3.4 Feature Selection
We generated and experimented with many features. We describe here only those features used directly or indirectly by our three submitted runs which we describe in § 4. We used word2vec representation and WordNet antonym and synonym for word similarity unless anything else is mentioned specifically.

1. \{w2v_wa, ppdb_wa, ppdb_wa_penWa\}: similarity scores generated using word alignment based methods (penWa for scores penalized by unalignment score).
2. \{gmm\}: output of Gaussian Mixture Model.
3. \{dssm, cdssm\}: similarity scores using DSSM and CDSSM models (see § 3.2.4).
4. \{dssm_lr, skipthought_lr\}: similarity scores using logistic model with sentence representations from DSSM and skip-thought models (see § 3.2.5).
5. \{sim_vec\}: score using similarity vector method (see § 3.2.6).
6. \{res_vec\}: score using the weighted resultant vector method (see § 3.2.7).
7. \{interpretable\}: score calculated using interpretable similarity method (§ 3.2.2).
8. \{noun_wa, verb_wa, adj_wa, adv_wa\}: Noun-Noun, Adjective-Adjective, Adverb-Adverb, and Verb-Verb alignment scores using word2vec for word similarity.
9. \{noun_verb_mult\}: multiplication of Noun-Noun similarity scores and Verb-Verb similarity scores.
10. \{abs_diff\}: absolute difference as \(\frac{|C_{ta} - C_{tb}|}{C_{ta} + C_{tb}}\) where \(C_{ta}\) and \(C_{tb}\) are the counts of tokens of type \(t\) \(\in\{\text{all tokens, adjectives, adverbs, nouns, and verbs}\}\) in sentence \(A\) and \(B\) respectively.
11. \{overlap_pen\}: unigram overlap between text \(A\) and \(B\) with synonym check given by:

$$score = \frac{2\times\text{overlap count}}{|A| + |B|}$$

Then penalized by crossing followed by unalignment score.
12. \{noali\}: number of NOALI relations in aligning chunks between texts relative to the total number of alignments (see § 3.2.2).
13. \{align, unalign\}: fraction of aligned/non-aligned words in the sentence pair.
14. \{mmr_t\}: min to max ratio as \(\frac{C_{t_1}}{C_{t_2}}\) where \(C_{t_1}\) and \(C_{t_2}\) are the counts of type \(t\) \(\in\{\text{all adjectives, adverbs, nouns, and verbs}\}\) for shorter text 1 and longer text 2 respectively.

4 Model Development

Training Data. We used data released in previous shared tasks (see Table 1) for the model development (see § 5 for STS benchmarking).

Models and Runs. Using the combination of features described in § 3.4, we built three different models corresponding to the three runs (R1-3) submitted.

R1. Linear SVM Regression model (SVR; \(\epsilon = 0.1, C = 1.0\)) with a set of 7 features: overlap_pen, ppdb_wa_penWa, dssm, dssm_lr, noali, abs_diff_all_tkns, mmr_all_tkns.

R2. Linear regression model (LR; default weka settings) with a set of 8 features: dssm, cdssm, gmm, res_vec, skipthought_lr, sim_vec, aligned, noun_wa.

122
| Data set          | Count | Release time       |
|-------------------|-------|--------------------|
| Deft-news         | 299   | STS2014-Test       |
| Images            | 749   | STS2014-Test       |
| Images            | 750   | STS2015-Test       |
| Headlines         | 742   | STS2015-Test       |
| Answer-forums     | 375   | STS2015-Test       |
| Answer-students   | 750   | STS2015-Test       |
| Belief            | 375   | STS2015-Test       |
| Headlines         | 244   | STS2016-Test       |
| Plagiarism        | 230   | STS2016-Test       |
| **Total**         | **4514** |                |

Table 1: Summary of training data.

| R1    | R2    | R3    | Baseline | 1st  |
|-------|-------|-------|----------|------|
| 0.8536| 0.8360| 0.8329| 0.7278   | **0.8547** |

Table 2: Results of our submitted runs on test data (1st is the best result among the participants).

R3. Gradient boosted regression model (GBR; *estimators* = 1000, *max_depth* = 3) which includes 3 additional features: *w2v_wa, ppdb_wa, overlap* to feature set used in Run 2.

We used SVR and and LR models in Weka 3.6.8. We used GBR model using sklearn python library. We evaluated our models on training data using 10-fold cross validation. The correlation scores in the training data were 0.797, 0.816 and 0.845 for R1, R2, and R3, respectively.

5 Results

Table 2 presents the correlation (r) of our system outputs with human ratings in the evaluation data (250 sentence pairs from Stanford Natural Language Inference data (Bowman et al., 2015)). The correlation scores of all three runs are 0.83 or above, on par with top performing systems. All of our systems outperform the baseline by a large margin of above 10%. Interestingly, R1 system is at par with the 1st ranked system differing by a very small margin of 0.009 (<0.2%). Figure 1 presents the graph showing R1 system output against human judgments (gold scores). It shows that our system predicts relatively better for similarity scores between 3 to 5 while the system slightly overshoots the prediction for the gold ratings in the range of 0 to 2. In general, it can be seen that our system works well across all similarity levels.

Our 11 features had a correlation of 0.75 or above when compared with gold scores in test data. In Table 3, we list only those features having correlations of 0.8 or above. Similarity scores computed using word alignment and compositional sentence vector methods were the best predictive features.

STS Benchmark (Agirre et al., 2017). We also evaluated our models on a benchmark dataset which consists of 1379 pairs and was created by the task organizers. We trained our three runs with the benchmark training data under identical settings. We used benchmark development data only for generating features from § 3.2.5 (as validation dataset). The correlation scores for R1, R2 and R3 systems were:

In Dev: 0.800, 0.822, **0.830** and
In Test: 0.755, 0.787, **0.792**

All of our systems outperformed best baseline benchmark system (Dev = 0.77, Test = 0.72). Interestingly, R3 was the best performing while R1 was the least performing among the three. As such, generalization was found to improve with increasing number of features (#features: 7, 8 and 11 for R1, R2 and R3 respectively).

6 Conclusion

We presented our DT.Team system submitted in SemEval-2017 Task 1. We developed three different models using SVM regression, Linear regression and Gradient Boosted regression for predicting textual semantic similarity. Overall, the outputs of our models highly correlate (correlation up to 0.85 in STS 2017 test data and up to 0.792 on benchmark data) with human ratings. Indeed, our methods yielded highly competitive results.
References

Eneko Agirre, Daniel Cer, Mona Diabe, Inigo Lopez-Gazpioa, and Specia Lucia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diabe, Aitor Gonzalez-Agirre, Weiwei Guof, Inigo Lopez-Gazpioa, Montse Maritxalar, Rada Mihalceab, et al. 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), pages 252–263.

Rajendra Banjade, Nabin Maharjan, Dipesh Gautam, and Vasile Rus. 2016. Dtsim at semeval-2016 task 1: Semantic similarity model including multi-level alignment and vector-based compositional semantics. Proceedings of SemEval pages 640–644.

Rajendra Banjade, Nobal B Niraula, Nabin Maharjan, Vasile Rus, Dan Stefanescu, Mihai Lintean, and Dipesh Gautam. 2015. Nerosim: A system for measuring and interpreting semantic textual similarity. In Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), pages 164–171.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. arXiv preprint arXiv:1508.05326.

Jianfeng Gao, Li Deng, Michael Gamon, Xiaodong He, and Patrick Pantel. 2014. Modeling interestingness with deep neural networks. US Patent App. 14/304,863.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In Proceedings of the 22nd ACM international conference on Conference on information & knowledge management. ACM, pages 2333–2338.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. arXiv preprint arXiv:1506.06726.

Harold W Kuhn. 1955. The hungarian method for the assignment problem. Naval research logistics quarterly 2(1-2):83–97.

Nabin Maharjan, Rajendra Banjade, Nobal B Niraula, and Vasile Rus. 2016. Semaligner: A method and tool for aligning chunks with semantic relation types and semantic similarity scores. CRF 82:62–56.

Nabin Maharjan, Rajendra Banjade, and Vasile Rus. 2017. Automated assessment of open-ended student answers in tutorial dialogues using gaussian mixture models (in press). In FLAIRS Conference.