SimiHawk at SemEval-2016 Task 1: A Deep Ensemble System for Semantic Textual Similarity

Peter Potash*, **William Boag***, **Alexey Romanov***,
Vasili Ramanishka, Anna Rumshisky
Dept. of Computer Science
University of Massachusetts Lowell
198 Riverside St, Lowell, MA 01854
{ppotash,wboag,aromanov,vramanis,arum}@cs.uml.edu

Abstract

This paper describes the SimiHawk system submission from UMass Lowell for the core Semantic Textual Similarity task at SemEval-2016. We built four systems: a small feature-based system that leverages word alignment and machine translation quality evaluation metrics, two end-to-end LSTM-based systems, and an ensemble system. The LSTM-based systems used either a simple LSTM architecture or a Tree-LSTM structure. We found that of the three base systems, the feature-based model obtained the best results, outperforming each LSTM-based model's correlation by .06. Ultimately, the ensemble system was able to outperform the base systems substantially, obtaining a weighted Pearson correlation of 0.738, and placing 7th out of 115 participating systems. We find that the ensemble system's success comes largely from its ability to form a consensus and eliminate complementary noise from its base systems' predictions.

1 Introduction

The task of semantic textual similarity (STS) has been developed over the past several SemEval competitions with the idea of capturing the degree of similarity in the meaning conveyed by two snippets of text (usually two sentences). In that respect, the STS task can be seen as similar to such tasks as paraphrase detection (Xu et al., 2015), recognizing textual entailment (RTE) (Negri et al., 2012), and semantic relatedness (Marelli et al., 2014a). The STS task captures different gradations of similarity, rather than a binary decision, and it is symmetric, rather than directed, as is the case with RTE (Agirre et al., 2012). It also aims to capture a more general notion of semantic similarity that does not focus solely on the semantic relatedness derived through compositional processes. In this paper, we describe the SimiHawk system submission for the core Semantic Textual Similarity (STS) task at the SemEval-2016 competition.

The STS task series (Agirre et al., 2012; Agirre et al., 2013; Agirre et al., 2014; Agirrea et al., 2015) has aggregated a sizable dataset of sentence pairs annotated with numeric similarity scores. The presence of this dataset allows for a shift from earlier work that mostly used unsupervised learning (Corley and Mihalcea, 2005; Mihalcea et al., 2006; Li et al., 2006), to the supervised approaches that leverage the labeled data (Sultan et al., 2015; Han et al., 2015; Hänig et al., 2015). The availability of labeled data from different genres also allows for a clearer evaluation and a better comparison across different approaches.

Specifically, the task of semantic similarity is defined as follows: given an input of two sentences, output a real number in the range [0,5] where 0 means lowest and 5 means highest similarity. The top-performing system from last year's task (Sultan et al., 2015) relied heavily on a hand-engineered word alignment tool (Sultan et al., 2014) (achieving 5th place with the aligner alone), combined with dense word embeddings (Baroni et al., 2014) to create a two-feature regression model. Other top-performing systems (Han et al., 2015; Hänig et al.,

*These three authors contributed equally to this work.

Proceedings of SemEval-2016, pages 1325–1332,
San Diego, California, June 16-17, 2016. ©2016 Association for Computational Linguistics

2015) follow this trend of exploiting word alignment and embedding similarity across textual pairs.

Recent work (Bowman et al., 2014; Bowman et al., 2015b; Tai et al., 2015) has shown the effectiveness of deep learning end-to-end architectures using recurrent and recursive neural networks on tasks similar to semantic similarity, such as semantic relatedness (Tai et al., 2015), natural language inference (Bowman et al., 2015a), and textual entailment (Marelli et al., 2014b), providing state-of-the-art performance. Since all of these tasks evaluate the relationship between the semantics of two input sentences, it stands to reason that systems with such architecture may perform well on the more general task of semantic similarity.

In our submission, we are interested in comparing an approach using hand-engineered features against the deep RNN-based architectures with Long Short-Term Memory (LSTM) cells. We therefore implemented a feature-based system over a small number of heavily engineered features, and two LSTM-based systems: one that uses a simple LSTM architecture and one that uses a Tree-LSTM architecture that mirrors the syntactic dependencies of the input in the LSTM model.

In the following section, we describe each system in detail. We then report and analyze the outcome of this bake-off. Not surprisingly, the ensemble system performs the best, obtaining a weighted Pearson correlation of 0.738. Of the three base systems, the feature-based model obtained the best results, outperforming the LSTM-based models by .06.

2 System Description

2.1 Feature-Based

Because of the impressive success of feature-based methods in previous STS shared tasks, we wanted to understand whether deep learning approaches were even necessary. The winning system for the 2015 shared task computed two features:

1. **alignment ratio** As part of their 1st place system, (Sultan et al., 2015) constructed a custom, open tool to align the words within a sentence pair[1]. The alignment ratio feature r is computed as:

$$r = \frac{a(s_1) + a(s_2)}{t(s_1) + t(s_2)} \tag{1}$$

where s_1 and s_2 are sentences 1 and 2 respectively, $a()$ is the number of aligned content words in a sentence, and $t()$ is the total number of content words in a sentence.

2. **cosine of word2vec centroids** Since word2vec came out in 2013 (Mikolov et al., 2013), word embeddings have gained massive popularity. Though a naive form of compositionality, simple vector addition has been shown as a surprisingly effective way to represent phrases. Using the official word2vec skip-gram Google News embeddings[2], we compute the word2vec cosine feature w as follows:

$$w = \cos(\frac{\Sigma w_1^i}{N_1}, \frac{\Sigma w_2^i}{N_2}) \tag{2}$$

where Σw_s^i is the sum of the embeddings of each word in sentence s and N_s is the sentence length.

We used this system as a starting point, but we added two additional feature classes. The two feature classes we added were:

1. **cosine of one-hot bag-of-words** Similar to word2vec centroids, we also computed a feature b as the cosine of binary bag-of-words encodings of each sentence:

$$b = \cos(\Sigma e_1^i, \Sigma e_2^j) \tag{3}$$

where for a vocabulary (consisting of the union of the two sentences' sets of tokens) of size V, the one-hot vector e_s^i is a vector of size V consisting of $V - 1$ zero entries and a 1 at word i's entry in the vocabulary. Note that subscripts 1 and 2 signify one-hot vectors representing tokens from sentences 1 and 2, respectively.

2. **Machine Translation evaluation metrics** In 2012, an ensemble of various MT metrics was created to attain (at the time) state-of-the-art results in Paraphrase Identification (Madnani et

[1] https://github.com/ ma-sultan/monolingual-word-aligner

[2] https://code.google.com/archive/p/word2vec

al., 2012). We created features for each of the following metrics to compute the similarity between sentences 1 and 2:

- BLEU (Papineni et al., 2002)[3]
- NIST (Doddington, 2002)[4]
- METEOR (Lavie and Agarwal, 2005)[5]
- BADGER (Parker, 2008)[6]
- TER (Snover et al., 2006)[7]
- TERp (Snover et al., 2009)[8]

We combine these four feature classes using a fully-connected neural network with 2 layers of size 40. Similar to (Tai et al., 2015) (specifically equations (14) and (15)), the network produces a probability distribution over scores and is trained using categorical cross-entropy loss. To build the network, we use the Keras library (Chollet, 2015), a Python neural network library written on top of Theano (Bergstra et al., 2010; Bastien et al., 2012).

2.2 Deep End-to-End LSTM-Based Systems

Deep end-to-end systems are very enticing because they learn the representation for a given input, as opposed to manually constructing a feature space. We built two LSTM-based systems for this task. Although in principle these systems can be considered end-to-end, both systems make use of pre-trained word embeddings as input. We describe the two systems below, assuming a familiarity with regular LSTM cell structure. We begin by discussing Tree-LSTM because it recently achieved state-of-the-art performance on the semantic relatedness task (Tai et al., 2015).

2.2.1 Tree-LSTM

The architecture of Tree-LSTM is a generalization of LSTMs to tree-structured network topologies. The tree-structured LSTM composes its current state from an input vector, as well as the hidden states of an arbitrary number of child units. As previously mentioned, the Tree-LSTM architecture has been used for the semantic relatedness task. For this task, the specific tree structure the authors use is a dependency tree, obtained via the Stanford Neural Network Dependency Parser (Chen and Manning, 2014). For the semantic relatedness task, the system takes as input two pieces of text, and outputs a real number in the range [1,5]. Therefore, one only needs to modify the original Tree-LSTM system to output a number in the range [0,5] in order to apply it to the task of semantic similarity.

The specific architecture we use is referred to by the original authors as Child-Sum Tree-LSTM (Tai et al., 2015). Using the authors' original notation, and using $C(j)$ to denote the set of children of node j, the hidden state at a given node is computed as follows:

$$\tilde{h_j} = \sum_{k \in C(j)} h_k \tag{4}$$

$$i_j = \sigma(W^i x_j + U^i \tilde{h_j} + b^i) \tag{5}$$

$$f_{j_k} = \sigma(W^f x_j + U^f h_k + b^f) \tag{6}$$

$$o_j = \sigma(W^o x_j + U^o \tilde{h_j} + b^o) \tag{7}$$

$$u_j = tanh(W^u x_j + U^u \tilde{h_j} + b^u) \tag{8}$$

$$c_j = i_j \odot u_j + \sum_{k \in C(j)} f_{j_k} \odot u_k \tag{9}$$

$$h_j = o_j \odot tanh(c_j) \tag{10}$$

where x_j is the input of node j (typically a word embedding vector), σ is the logistic sigmoid function, and $\odot$ is element-wise multiplication. Note here we refer to nodes instead of timesteps, since the the tree structure induced by the dependency parse alters the standard temporal flow of a normal LSTM. Consequently, the hidden state from the root node is then used as a representation for a given text input.

Given two input texts, the authors use a Tree-LSTM to generate a representation for each input, which we call h_L and h_R. These are then used to compute a final score, $\hat{y}$, as follows:

$$h_\times = h_L \odot h_R \tag{11}$$

$$h_+ = |h_L - h_R| \tag{12}$$

$$h_s = \sigma(W^\times h_\times + W^+ h_+ + b^h) \tag{13}$$

[3] ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v13a-20091001.tar.gz

[4] ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v13a-20091001.tar.gz

[5] http://www.cs.cmu.edu/ alavie/METEOR/download/meteor-1.5.tar.gz

[6] http://babblequest.org/badger

[7] http://www.cs.umd.edu/˜snover/tercom/tercom-0.7.25.tgz

[8] https://github.com/snover/terp

$$\hat{p_\theta} = softmax(W^p h_s + b^p) \quad (14)$$

$$\hat{y} = r^T \hat{p_\theta} \quad (15)$$

Note how (14) creates a distribution over integer scores, and (15) converts the distribution into a real number in range $[a,b]$ by setting r equal to a vector of integer values $[a \ ... \ b\ ]$. For our system, we simply set r equal to the vector [0 1 2 3 4 5], as opposed to the original values of [1 2 3 4 5] for the semantic relatedness task. The inclusion of zero will better facilitate the output of scores less than one. This is because distributions that have their mass highly concentrated in the first index will have little remaining mass in the other indices that can then multiply by the non-zero integers.

For our implementation we used the code from the original authors[9], which uses Glove embeddings (Pennington et al., 2014) as input vectors x_j, hidden states (h_i) of size 200, and a similarity module of size 50 (h_s (13)). For training, we use a regularization strength of 0.0004, a minibatch size of 25, and a learning rate of 0.05. The model trained for 10 epochs. We also decided to disregard the use of a validation set, since our intermediate results showed that, if anything, the use of a validation set hurt performance.

2.2.2 LSTM

Despite the potential of the Tree-LSTM model for this task, (Bowman et al., 2015b) shows that even for text that is highly syntactically dependent, a standard LSTM architecture can perform as well as a Tree-LSTM architecture given enough training data. The authors also show that an LSTM can perform better on shorter sentences. Moreover, (Vinyals et al., 2015) successfully applied a sequence-to-sequence LSTM model with attention to the syntactic parsing task, achieving state-of-the-art results. Motivated by these results, we also produced a model that uses standard LSTM architecture.

The architecture of the LSTM model is represented in Figure 1. The model is very similar to the Tree-LSTM, but instead of using tree-structured LSTMs, it uses standard LSTM cells. The hidden states of the LSTMs are combined by concatenation of element-wise multiplication, summation and cosine similarity, and the resulting vector serves as the input to the fully-connected layer. As in (Tai et al., 2015), the network produces a probability distribution over scores, which is in turn transformed into a real number over the specified range.

We used the Keras library to implement this model, creating LSTMs with hidden states of size 300, and fully-connected layers of size 50. For training, we use a regularization strength of 0.0004, minibatch size of 25, and dropout of 0.1. The model was trained for 15 epochs. We also trimmed sentences to have a maximum length of 50 tokens. The LSTM system uses Glove embeddings (Pennington et al., 2014) as its pretrained word vectors. We chose to use Glove instead of word2vec because Glove had fewer out-of-vocabulary tokens for the dataset.

2.3 Ensemble

In order to hedge our bets, we created a ensemble system, using Feature-Based, Tree-LSTM, and LSTM systems as base systems. The ensemble model can be seen as a feature-based, stacking system because it uses the predictions of the three base systems as the features for a supervised model. Thus, the most basic stacking system has three features, one for each base system.

In order to train the stacking model, we need base predictions that are not 'dirty': the data used to train base systems to generate predictions (which in turn train our stacking system) must have an empty intersection with the data we use to make predictions. To overcome this, each base system performed 5-fold cross-validation on the training data. By doing this, we create predictions for each training example from 'clean' data. Next, we used the predictions on the held-out folds as the features to train the stacking model. We made sure to shuffle the training data prior to creating the folds to ensure that each fold would be statistically similar in terms of which domains are represented as well as the distributions over gold standard scores. To generate test scores for the stacking system, we allowed the base systems to train on all the training data and then make predictions on the test data, which become the test features for the stacking system.

We hypothesize that in certain scenarios, the stacking system should favor the scores from certain

[9] https://github.com/stanfordnlp/treelstm

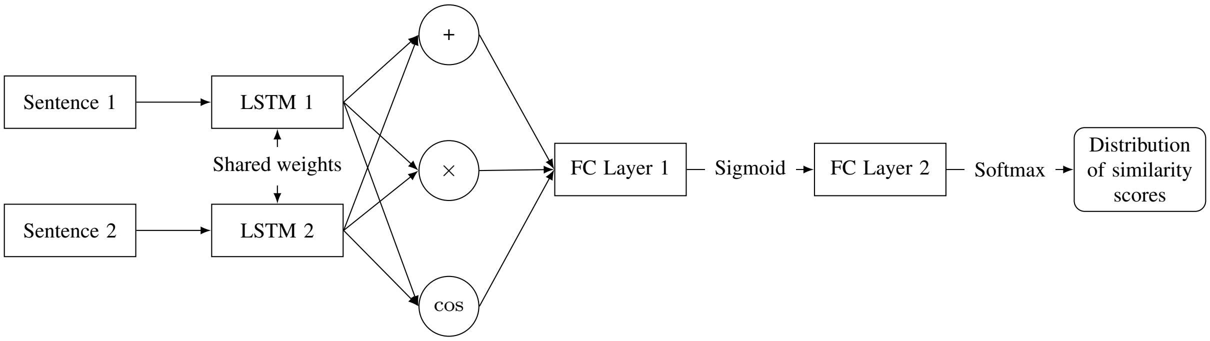

Figure 1: The architecture of the LSTM model

System	mean	answer-answer	headlines	plagiarism	postediting	question-question
Ensemble	**0.73774**	**0.59237**	**0.81419**	0.80566	**0.82179**	0.65048
Feature-based	0.70647	0.44003	0.77109	**0.81105**	0.81600	**0.71035**
LSTM	0.64840	0.44177	0.75703	0.71737	0.72317	0.60691
TreeLSTM	0.64140	0.52277	0.74083	0.67628	0.70655	0.55265
# Examples	-	254	249	230	244	209

Table 1: Results of our systems on the 2016 gold standard. The highest score from each column is in bold.

Features	Weighted Mean
All	0.7168
-average pronoun count	0.7178
-average length	0.7211
-embeddings similarity	0.7237
-edit distance	0.7164
-align ratio	0.6952
-sentence length	0.7116

Table 2: Feature ablation study for ensemble system. '-' denotes witholding a feature.

base systems. For example, (Bowman et al., 2015b) has shown that LSTM models can have difficulty on long sequences. Therefore, we performed a feature ablation study where we combined the three baseline predictions with several hand-picked features: length of sentence 1, length of sentence 2 (these two combined are called sentence length), alignment ratio (1), string edit distance, embedding similarity (2), average sentence length between the pair, and average pronoun count. We included average pronoun count because we found certain domains contain a larger frequency of pronouns. The ablation experiment was performed on 2015 evaluation data, and the results are shown in Table 2. It is visible in the table that the only feature absence that generated any significant drop-off is alignment ratio. Therefore, we decided to include alignment ratio as the fourth and final feature in our stacking system. For the stacking implementation, we use the same neural network as the feature-based system. However, we decided to use a three-layer network based on experimental results.

3 Results

All systems were trained using the training/evaluation data from previous years' tasks (Agirre et al., 2012; Agirre et al., 2013; Agirre et al., 2014; Agirrea et al., 2015). After filtering for duplicate examples, our training set contains a total of 13,061 examples.

Results of our systems on the 2016 gold standard are shown in Table 1. We report the performance, specifically Pearson correlation, across five different domains: Answer-Answer, Headlines, Plagiarism, Postediting, and Question-Question. We also report the weighted mean across all five domains. The last row of Table 1 also records the number of gold standard examples for each domain.

According to the official results, the overall performance of our ensemble system was .738, exceeding the performance of the base systems, although it is not the highest performing system for every domain. Specifically for the question-question domain, the feature-based system seems to outperform it substantially. The overall ranking of the ensemble system is 7 out of 115 submitted systems, while feature-based ranks 37, LSTM ranks 73, and TreeLSTM ranks 77.

System	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	mean
LSTM	0.802	0.806	0.805	0.795	0.790	0.800
Feature-based	0.790	0.800	0.813	0.823	0.799	0.805
TreeLSTM	0.772	0.797	0.793	0.806	0.794	0.792

Table 3: Results of our base systems on cross-validation folds

System	Ensemble	Features	LSTM	TreeLSTM	Reference
Ensemble	1.000	0.769	0.751	0.802	0.592
Feature-based	0.769	1.000	0.456	0.413	0.440
LSTM	0.751	0.456	1.000	0.608	0.442
TreeLSTM	0.802	0.413	0.608	1.000	0.523
Reference	0.592	0.440	0.442	0.523	1.000

Table 4: Pairwise correlation scores amongst systems' predictions and gold labels (Reference) on the answer-answer domain.

We also report five-fold cross-validation results for our three base systems (Table 3). These experiments used the training data, drawn from several domains. We shuffled the data for our cross-validation experiments so that each fold has a similar distribution of domain examples as well as scores. These results show that when the data is shuffled, and domains and scores are equally distributed, the overall correlation is much higher.

4 Discussion

The ensemble system's success can be attributed to its ability to form a consensus among the base systems and eliminate noise in the predictions. As we can see in Table 4, which shows the correlation between systems for the answer-answer domain, although the three base systems have low pairwise correlations (.456, .413, .608), they all have high correlations with the ensemble system (.769, .751, .802). We can also see that the ensemble system has its largest correlation with the highest-scoring base system for answer-answer, TreeLSTM. Since each base system had pairwise low correlation, they were capturing different views of the data, and the ensemble system was able to take the important parts of each view, while canceling out the noise. This allowed the ensemble system to outperform the base systems in answer-answer by .12, on average.

To further illustrate this point, take for example the following input pair: "There's not a lot you can do about that." and "There's not that much that you can do with a sourdough starter." with a gold label of 2. The baseline systems predict the following scores: 3.96 from the feature-based system, 0.31 from the LSTM system, and 1.39 from the TreeLSTM system. The TreeLSTM system provides the closest prediction, while the feature-based system provides the worst prediction, over-predicting the gold label by almost 2. However, the ensemble system is able to effectively leverage this higher prediction, producing a prediction of 1.76, which is the most accurate prediction out of all four systems.

In our experiments, we found that in three domains (answer-answer, headlines, question-question), the ensemble system had the highest correlation with TreeLSTM. It is likely that TreeLSTM's low score on question-question (.553), combined with a high correlation with the ensemble (.900), were what brought the ensemble system's score down enough for the feature-based system to outperform it. On the other two occasions (plagiarism and postediting), the base system with the highest correlation with ensemble system was the feature-based system. In both of these scenarios, the two systems produced very similar scores, within .01 of one another for both occasions.

Although the feature-based system was the highest scoring base system on 4 out of 5 domains, there is only one occasion where its correlation with another base system exceeds .79 (plagiarism with LSTM). This suggests that the shallow methods for feature extraction are not able to represent some important information that the deep LSTM-based models are able to capture. As a result, the ensemble system was able to outperform all three base systems in most cases, suggesting that none of the base systems were able to capture the whole picture individually.

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 Weiwei Guo. 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*. Citeseer.

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.

Eneko Agirrea, Carmen Baneab, Claire Cardiec, Daniel Cerd, Mona Diabe, Aitor Gonzalez-Agirrea, Weiwei Guof, Inigo Lopez-Gazpioa, Montse Maritxalara, 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.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL (1)*, pages 238–247.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a cpu and gpu math expression compiler. In *Proceedings of the Python for scientific computing conference (SciPy)*, volume 4, page 3. Austin, TX.

Samuel R Bowman, Christopher Potts, and Christopher D Manning. 2014. Recursive neural networks can learn logical semantics. *arXiv preprint arXiv:1406.1827*.

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

Samuel R Bowman, Christopher D Manning, and Christopher Potts. 2015b. Tree-structured composition in neural networks without tree-structured architectures. *arXiv preprint arXiv:1506.04834*.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*, pages 740–750.

Franois Chollet. 2015. Keras. https://github.com/fchollet/keras.

Courtney Corley and Rada Mihalcea. 2005. Measuring the semantic similarity of texts. In *Proceedings of the ACL workshop on empirical modeling of semantic equivalence and entailment*, pages 13–18. Association for Computational Linguistics.

G. Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of HLT*, pages 138–145.

Lushan Han, Justin Martineau, Doreen Cheng, and Christopher Thomas. 2015. Samsung: Align-and-differentiate approach to semantic textual similarity. *SemEval-2015*, page 172.

Christian Hänig, Robert Remus, and Xose De La Puente. 2015. Exb themis: Extensive feature extraction from word alignments for semantic textual similarity. *SemEval-2015*, page 264.

Alon Lavie and Abhaya Agarwal. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL 2005 Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*, pages 65–72.

Yuhua Li, David McLean, Zuhair A Bandar, James D O'shea, and Keeley Crockett. 2006. Sentence similarity based on semantic nets and corpus statistics. *Knowledge and Data Engineering, IEEE Transactions on*, 18(8):1138–1150.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 182–190, Montréal, Canada, June. Association for Computational Linguistics.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014a. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *SemEval-2014*.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014b. A sick cure for the evaluation of compositional distributional semantic models. In *LREC*, pages 216–223.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, volume 6, pages 775–780.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Matteo Negri, Alessandro Marchetti, Yashar Mehdad, Luisa Bentivogli, and Danilo Giampiccolo. 2012. Semeval-2012 task 8: cross-lingual textual entailment for content synchronization. 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 399–407. Association for Computational Linguistics.

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

Steven Parker. 2008. Badger: A new machine translation metric.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Matthew G. Snover, Nitin Madnani, Bonnie Dorr, and Richard Schwartz. 2009. Ter-plus: Paraphrase, semantic, and alignment enhancements to translation edit rate.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2014. Back to basics for monolingual alignment: Exploiting word similarity and contextual evidence. *Transactions of the Association for Computational Linguistics*, 2:219–230.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2015. Dls@ cu: Sentence similarity from word alignment and semantic vector composition. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 148–153.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2755–2763.

Wei Xu, Chris Callison-Burch, and William B Dolan. 2015. Semeval-2015 task 1: Paraphrase and semantic similarity in twitter (pit). *Proceedings of SemEval*.