Stanford: Probabilistic Edit Distance Metrics for STS

Mengqiu Wang and **Daniel Cer**[*]
Computer Science Department
Stanford University
Stanford, CA 94305 USA
{mengqiu,danielcer}@cs.stanford.edu

Abstract

This paper describes Stanford University's submission to SemEval 2012 Semantic Textual Similarity (STS) shared evaluation task. Our proposed metric computes probabilistic edit distance as predictions of semantic similarity. We learn weighted edit distance in a probabilistic finite state machine (pFSM) model, where state transitions correspond to edit operations. While standard edit distance models cannot capture long-distance word swapping or cross alignments, we rectify these shortcomings using a novel pushdown automaton extension of the pFSM model. Our models are trained in a regression framework, and can easily incorporate a rich set of linguistic features. The performance of our edit distance based models is contrasted with an adaptation of the Stanford textual entailment system to the STS task. Our results show that the most advanced edit distance model, pPDA, outperforms our entailment system on all but one of the genres included in the STS task.

1 Introduction

We describe a probabilistic edit distance based metric, which was originally designed for evaluating machine translation quality, for computing semantic textual similarity (STS). This metric models weighted edit distance in a probabilistic finite state machine (pFSM), where state transitions correspond to edit operations. The weights of the edit operations are automatically learned in a regression framework. One of the major contributions of this paper is a novel extension of the pFSM model into a probabilistic Pushdown Automaton (pPDA), which enhances traditional edit-distance models with the ability to model phrase shift and word swapping. Furthermore, we give a new log-linear parameterization to the pFSM model, which allows it to easily incorporate rich linguistic features. We contrast the performance of our probabilistic edit distance metric with an adaptation of the Stanford textual entailment system to the STS task.

[*] Daniel Cer is one of the organizers for the STS task. The STS test set data was not used in any way for the development or training of the systems described in this paper.

2 pFSMs for Semantic Textual Similarity

We start off by framing the problem of semantic textual similarity in terms of weighted edit distance calculated using probabilistic finite state machines (pFSMs). A FSM defines a language by accepting a string of input tokens in the language, and rejecting those that are not. A probabilistic FSM defines the probability that a string is in a language, extending on the concept of a FSM. Commonly used models such as HMMs, *n*-gram models, Markov Chains and probabilistic finite state transducers all fall in the broad family of pFSMs (Knight and Al-Onaizan, 1998; Eisner, 2002; Kumar and Byrne, 2003; Vidal et al., 2005). Unlike all the other applications of FSMs where tokens in the language are words, in our language tokens are edit operations. A string of tokens that our FSM accepts is an edit sequence that transforms one side of the sentence pair (denoted as $\mathbf{s^1}$) into the other side ($\mathbf{s^2}$).

Our pFSM has a unique start and stop state, and one state per edit operation (i.e., *Insert*, *Delete*, *Substitution*). The probability of an edit sequence **e** is generated by the model is the product of the state transition probabilities in the pFSM, formally described

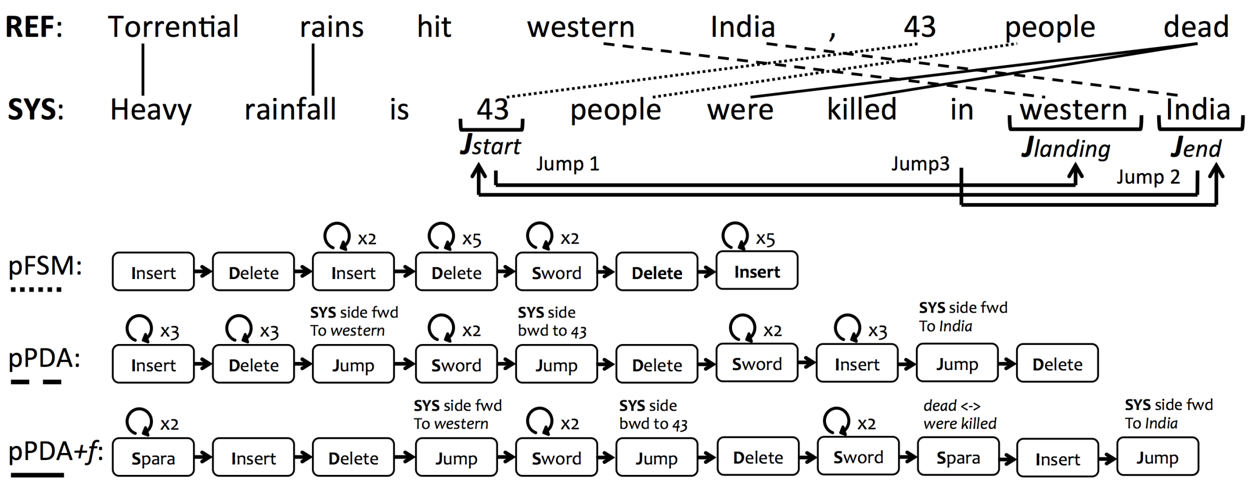
REF: Torrential rains hit western India , 43 people dead
SYS: Heavy rainfall is 43 people were killed in western India
Jstart
Jlanding
Jend
Jump 1
Jump3
Jump 2
pFSM:
Insert Delete Insert x2 Delete x5 Sword x2 Delete Insert x5
pPDA:
Insert x3 Delete x3 Jump Sword x2 Jump Delete Sword x2 Insert x3 Jump Delete
SYS side fwd To western
SYS side bwd to 43
SYS side fwd To India
pPDA+f:
Spara x2 Insert Delete Jump Sword x2 Jump Delete Sword x2 Spara Insert Jump
dead <-> were killed

Figure 1: This diagram illustrates an example sentence pair from the statistical machine translation subtask of STS. The three rows below are the best state transition (edit) sequences that transforms REF($\mathbf{s^1}$) to SYS($\mathbf{s^2}$). The corresponding alignments generated by the models (pFSM, pPDA, pPDA+*f*) are shown with different styled lines, with later models in the order generating strictly more alignments than earlier ones. The gold human evaluation score is 6.5, and model predictions are: pPDA+f 5.5, pPDA 4.3, pFSM 3.1.

as:

$$w(\mathbf{e} \mid \mathbf{s^1},\mathbf{s^2}) = \frac{1}{Z}\prod_{i=1}^{|\mathbf{e}|} \exp \theta \cdot \mathbf{f}(e_{i-1},e_i,\mathbf{s^1},\mathbf{s^2}) \quad (1)$$

We featurize each of the state changes with a log-linear parameterization; **f** is a set of binary feature functions defined over pairs of neighboring states (by the Markov assumption) and the input sentences, and θ are the associated feature weights; Z is a partition function. In this basic pFSM model, the feature functions are simply identity functions that emit the current state, and the state transition sequence of the previous state and the current state.

The feature weights are then automatically learned by training a global regression model where the human judgment score for each sentence pair is the regression target ($\hat{y}$). Since the "gold" edit sequence are not given at training or prediction time, we treat the edit sequences as hidden variables and sum over them in our model. We introduce a new regression variable $y \in \mathbb{R}$ which is the log-sum of the unnormalized weights (Eqn. (1)) of all edit sequences, formally expressed as:

$$y = \log \sum_{\mathbf{e}' \subseteq \mathbf{e}^*} \prod_{i=1}^{|\mathbf{e}'|} \exp \theta \cdot \mathbf{f}(e_{i-1},e_i,\mathbf{s^1},\mathbf{s^2}) \quad (2)$$

$\mathbf{e}^*$ is the set of all possible alignments. The sum over an exponential number of edit sequences in $\mathbf{e}^*$ is solved efficiently using a forward-backward style dynamic program. Any edit sequence that does not lead to a complete transformation of the sentence pair has a probability of zero in our model. Our regression target then seeks to minimize the least squares error with respect to $\hat{y}$, plus a $L2$-norm regularizer term parameterized by λ:

$$\theta^* = \min_{\theta} \{ \sum_{\mathbf{s^1_i},\mathbf{s^2_i}} [\hat{y}_i - (\frac{y}{|\mathbf{s^1_i}|+|\mathbf{s^2_i}|} + \alpha)]^2 + \lambda \|\theta\|^2 \} \quad (3)$$

The $|\mathbf{s^1_i}|+|\mathbf{s^2_i}|$ is a length normalization term for the ith training instance, and α is a scaling constant whose value is to be learned. At test time, $y/(|\mathbf{s^1}|+|\mathbf{s^2}|)+\alpha$ is computed as the predicted score.

We replaced the standard substitution edit operation with three new operations: S_{word} for same word substitution, S_{lemma} for same lemma substitution, and S_{punc} for same punctuation substitution. In other words, all but the three matching-based substitutions are disallowed. The start state can transition into any of the edit states with a constant unit cost, and each edit state can transition into any other edit state if and only if the edit operation involved is valid at the current edit position (e.g., the model cannot transition into *Delete* state if it is already at the end of

$\mathbf{s^1}$; similarly it cannot transition into S_{lemma} unless the lemma of the two words under edit in $\mathbf{s^1}$ and $\mathbf{s^2}$ match). When the end of both sentences are reached, the model transitions into the stop state and ends the edit sequence. The first row in Figure 1 starting with pFSM shows a state transition sequence for an example sentence pair. There exists a one-to-one correspondence between substitution edits and word alignments. Therefore this example state transition sequence correctly generates an alignment for the word *43* and *people*.

2.1 pPDA Extension

A shortcoming of edit distance models is that they cannot handle long-distance word swapping — a pervasive phenomenon found in most natural languages. [1] Edit operations in standard edit distance models need to obey strict incremental order in their edit position, in order to admit efficient dynamic programming solutions. The same limitation is shared by our pFSM model, where the Markov assumption is made based on the incremental order of edit positions. Although there is no known solution to the general problem of computing edit distance where long-distance swapping is permitted (Dombb et al., 2010), approximate algorithms do exist. We present a simple but novel extension of the pFSM model to a probabilistic pushdown automaton (pPDA), to capture non-nested word swapping within limited distance, which covers a majority of word swapping in observed in real data (Wu, 2010).

A pPDA, in its simplest form, is a pFSM where each control state is equipped with a stack (Esparza and Kucera, 2005). The addition of stacks for each transition state endows the machine with memory, extending its expressiveness beyond that of context-free formalisms. By construction, at any stage in a normal edit sequence, the pPDA model can "jump" forward within a fixed distance (controlled by a max distance parameter) to a new edit position on either side of the sentence pair, and start a new edit subsequence from there. Assuming the jump was made on the $\mathbf{s^2}$ side, [2] the machine remembers its current edit position in $\mathbf{s^2}$ as J_{start}, and the destination position on $\mathbf{s^2}$ after the jump as $J_{landing}$.

We constrain our model so that the only edit operations that are allowed immediately following a "jump" are from the set of substitution operations (e.g., S_{word}). And after at least one substitution has been made, the device can now "jump" back to J_{start}, remembering the current edit position as J_{end}. Another constraint here is that after the backward "jump", all edit operations are permitted except for *Delete*, which cannot take place until at least one substitution has been made. When the edit sequence advances to position $J_{landing}$, the only operation allowed at that point is another "jump" forward operation to position J_{end}, at which point we also clear all memory about jump positions and reset.

An intuitive explanation is that when pPDA makes the first forward jump, a gap is left in $\mathbf{s^2}$ that has not been edited yet. It remembers where it left off, and comes back to it after some substitutions have been made to complete the edit sequence. The second row in Figure 1 (starting with pPDA) illustrates an edit sequence in a pPDA model that involves three "jump" operations, which are annotated and indexed by number 1-3 in the example. "Jump 1" creates an un-edited gap between word *43* and *western*, after two substitutions, the model makes "jump 2" to go back and edit the gap. The only edit permitted immediately after "jump 2" is deleting the comma in $\mathbf{s^1}$, since inserting the word *43* in $\mathbf{s^2}$ before any substitution is disallowed. Once the gap is completed, the model resumes at position J_{end} by making "jump 3", and completes the jump sequence.

The "jumps" allowed the model to align words such as *western India*, in addition to the alignments of *43 people* found by the pFSM. In practice, we found that our extension gives a big boost to model performance (*cf.* Section 4), with only a modest increase in computation time. [3]

2.2 Parameter Estimation

Since the least squares operator preserves convexity, and the inner log-sum-exponential function is convex, the resulting objective function is also convex.

[1]The edit distance algorithm described in Cormen et al. (2001) can only handle adjacent word swapping (transposition), but not long-distance swapping.

[2]Recall that we transform $\mathbf{s^1}$ into $\mathbf{s^2}$, and thus on the $\mathbf{s^2}$ side, we can only insert but not delete. The argument applies equally to the case where the jump was made on the other side.

[3]The length of the longest edit sequence with jumps only increased by $0.5 * max(|\mathbf{s^1}|, |\mathbf{s^2}|)$ in the worst case, and by and large swapping is rare in comparison to basic edits.

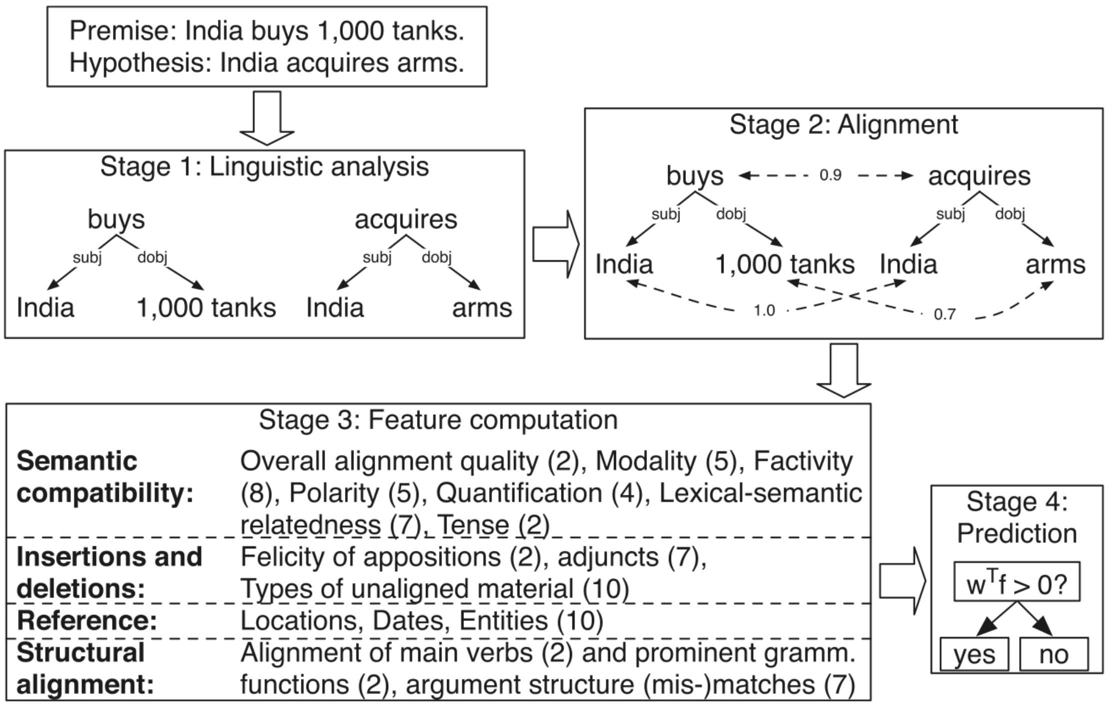
Premise: India buys 1,000 tanks.
Hypothesis: India acquires arms.
Stage 1: Linguistic analysis
buys
subj
dobj
India
1,000 tanks
acquires
India
arms
Stage 2: Alignment
0.9
1.0
0.7
Stage 3: Feature computation
Semantic compatibility:
Overall alignment quality (2), Modality (5), Factivity (8), Polarity (5), Quantification (4), Lexical-semantic relatedness (7), Tense (2)
Insertions and deletions:
Felicity of appositions (2), adjuncts (7), Types of unaligned material (10)
Reference:
Locations, Dates, Entities (10)
Structural alignment:
Alignment of main verbs (2) and prominent gramm. functions (2), argument structure (mis-)matches (7)
Stage 4: Prediction
w^Tf > 0?
yes
no

Figure 2: **Stanford Entailment Recognizer:** The pipelined approach used by the Stanford entailment recognizer to analyze sentence pairs and determine whether or not an entailment relationship is present. The entailment recognizer first obtains dependency parses for both the passage and the hypothesis. These parses are then aligned based upon lexical and structural similarity between the two dependency graphs. From the aligned graphs, features are extracted that suggest the presence or absence of an entailment relationship. Figure courtesy of (Pado et al., 2009).

For parameter learning, we used the limited memory quasi-newton method (Liu and Nocedal, 1989) to find the optimal feature weights and scaling constant for the objective. We initialized $\theta = \vec{0}$, $\alpha = 0$, and $\lambda = 5$. We also threw away features occurring fewer than five times in training corpus. Gradient calculation was similar to other pFSM models, such as HMMs, we omitted the details here, for brevity.

2.3 Rich Linguistic Features

We add new substitution operations beyond those introduced in Section 2, to capture synonyms and paraphrase in the sentence pair. Synonym relations are defined according to WordNet (Miller et al., 1990), and paraphrase matches are given by a lookup table. To better take advantage of paraphrase information at the multi-word phrase level, we extended our substitution operations to match longer phrases by adding one-to-many and many-to-many bigram block substitutions. In our experiments on machine translation evaluation task, which our metric was originally developed for, we found that most of the gain came from unigrams and bigrams, with little to no additional gains from trigrams. Therefore, we limited our experiments to bigram pFSM and pPDA models, and pruned the paraphrase table adopted from TERplus [4] to unigrams and bigrams, resulting in 2.5 million paraphrase pairs. Trained on all available training data, the resulting pPDA model has a total of 218 features.

2.4 Model Configuration

We evaluate both the pFSM and pPDA models with the addition of rich linguistic features, as described in the previous section. For pPDA model, the jump distance is set to five. For each model, we experimented with two different training schemes. In the first scheme, we train a separate model for each section of the training dataset (i.e., MSRpar, MSRvid, and SMTeuroparl), and use that model to test on their respective test set. For the two unseen test sets (SMT-

[4]Available from `www.umiacs.umd.edu/~snover/terp`.

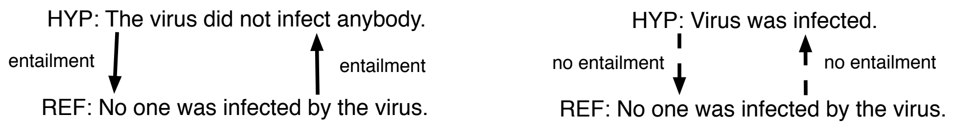
HYP: The virus did not infect anybody.
entailment
entailment
REF: No one was infected by the virus.
HYP: Virus was infected.
no entailment
no entailment
REF: No one was infected by the virus.

Figure 3: Semantic similarity as determined by mutual textual entailment. Figure courtesy of (Pado et al., 2009).

news and OnWN), we used a joint model trained on all of the available training data. We refer to this scheme as *Indi* henceforth. In the second scheme, we used the joint model trained on all training data to make preditions for all test sets (we refer to this scheme as *All*). Our official submission contains two runs – pFSM with scheme *Indi*, and pPDA with scheme *All*.

3 Textual Entailment for STS

We contrast the performance of the probabilistic edit distance metrics with an adaptation of the Stanford Entailment Recognizer to the STS task. In this section, we review the textual entailment task, the operation of the Stanford Entailment Recognizer, and describe how we adapted our entailment system to the STS task.

3.1 Recognizing Textual Entailment

The Recognizing Textual Entailment (RTE) task (Dagan et al., 2005) involves determining whether the meaning of one text can be inferred from another. The text providing the ground truth for the evaluation is known as the passage while the text being tested for entailment is known as the the hypothesis. A passage entails a hypothesis if a casual speaker would consider the inference to be correct. This intentionally side-steps strict logical entailment and implicitly brings in all of the world knowledge speakers use to interpret language.

The STS task and RTE differ in two significant ways. First, the RTE task is one directional. If a hypothesis sentence is implied by a passage, the inverse does not necessarily hold (e.g., “John is outside in the snow without a coat.” casually implies “John is cold”, but not vice versa). Second, the RTE task forces systems to make a boolean choice about entailment, rather than the graded scale of semantic relatedness implied by STS.

3.2 Textual Entailment System Description

Shown in Figure 2, the Stanford entailment system uses a linguistically rich multi-stage annotation pipeline. Incoming sentence pairs are first dependency parsed. The dependency parse trees are then transformed into semantic graphs containing additional annotations such as named entities and coreference. The two semantic graphs are then aligned based upon structural overlap and lexical semantic similarity using a variety of word similarity metrics based on WordNet, vector space distributional similarity as calculated by InfoMap, and a specialized module for matching ordinal values. The system then supplies the aligned semantic graphs as input to a number of feature producing modules. Some modules produce gross aggregate scores, such as returning the alignment quality between the two sentences. Others look for specific phenomena that suggest the presence or absence of an entailment relationship, such as a match or mismatch in polarity (e.g., “died” vs. “didn’t die”), tense, quantification, and argument structure. The resulting features are then passed on to a down stream classifier to predict whether or not an entailment relationship exists.

3.3 Adapting RTE to STS

In order to adapt our entailment recognition system to STS, we follow the same approach Pado et al. (2009) used to successfully adapt the entailment system to machine translation evaluation. As shown in Figure 3, for each pair of sentences presented to the system, we run the entailment system in both directions and extract features that describe whether the first sentence entails the second and vice versa for the opposite direction. This setup effectively treats the STS task as a bidirectional variant of the RTE task. The extracted bidirectional entailment features are then passed on to a support vector machine regression (SVR) model, which predicts the STS score for the sentence pair. As in Pado et al. (2009), we augment the bidirectional entailment features with sentence level BLEU

Models	All	MSRpar	MSRvid	SMTeuro	OnWn	SMTnews
pFSMIndi	$0.6354^{(38)}$	0.3795	0.5350	0.4377	-	-
pFSMAll	0.3727	0.3769	0.4569	0.4256	0.6052	0.4164
pPDAIndi	**0.6808**	0.4244	0.5051	0.4554	-	-
pPDAAll	$0.4229^{(77)}$	**0.4409**	0.4698	**0.4558**	**0.6468**	**0.4769**
Entailment	$0.5589^{(55)}$	0.4374	**0.8037**	0.3533	0.3077	0.3235

Table 1: Absolute score prediction results on STS12 test set. Numbers in this table are Pearson correlation scores. Best result on each test set is highlighted in bold. Numbers in *All* column that has superscript are the official submissions. Their relative ranks among 89 systems are shown in superscripts.

scores, in order to improve robustness over noisy non-grammatical data. We trained the SVR model using libSVM over all of the sentence pairs in the STS training set. The model uses a Gaussian kernel with $\gamma = 0.125$, an SVR ε-loss of 0.25, and margin violation cost, C, of 2.0. These hyperparameters were selected by cross validation over the training set.

4 Results

From Table 1, we can see that the pPDA model performed better than the pFSM model on all test sets except the MSRvid section. This result clearly demonstrates the power of the pPDA extension in modeling long-distance word swapping. The MSRvid test set has the shortest overall sentence length (13, versus 35 for MSRpar), and therefore it is not too surprising that long distance word swapping did not help much here. Furthermore, the pPDA model shows a much more pronounced performance gain than pFSM when tested on unseen datasets (OnWn and SMTnews), suggesting that the pPDA model is more robust across domain. A second observation is that the *Indi* training scheme seems to work better than the *All* approach, which shows having more training data does not compensate the different characteristics of each training portion. Our best metric on all test set is the pPDAIndi model, with a Pearson's correlation score of 0.6808. If interpolated into the official submitted runs ranking, it would be placed at the 22nd place among 89 runs. Among the three official runs submitted to the shared task (pPDAAll, pFSMIndi and Entailment), pFSMIndi performs the best, placed at 38th place among 89 runs. Since our metrics were originally designed for statistical machine translation (MT) evaluation, we found that on the unseen SMTNews test set, which consists of news conversation sentence pairs from the MT domain, our pPDA model placed at a much higher position (13 among 89 runs).

In comparison to results on MT evaluation task (Wang and Manning, 2012), we found that the pPDA and pFSM models work less well on STS. Whereas in MT evaluation it is common to have access to thousands of training examples, there is an order of magnitude less available training data in STS. Therefore, learning hundreds of feature parameters in our models from such few examples are likely to be ill-posed.

Overall, the RTE system did not perform as well as the regression based models except for the MSRvid domain , which has the shortest overall sentence length. Qualitative evaluation suggests that the MSRvid domain exhibits the least degree of lexical divergence between sentence pairs, thus making this task *easier* than other domains (the median score of all 89 official systems for MSRvid is 0.7538, while for the medians for MSRpar and SMTeuroparl are 0.5128 and 0.4437, respectively). The relative rank of RTE for MSRvid is 21 out of 89, whereas the pFSM and pPDA systems ranked 80 and 83, respectively. The low performance of pFSM and pPDA on this task significantly affected the overall rankings of these two systems. We do not have a clear explanation as to why RTE system thrives on this *easier* task while pPDA and pFSM seem to suffer. For future work, we aim to gain a better understanding of the different characteristics of these systems, and explore model combination techniques.

5 Conclusion

We describe a metric for computing sentence level semantic textual similarity, which is based on a probabilistic finite state machine model that computes weighted edit distance. Our model admits a rich set of linguistic features, and can be trained to learn feature weights automatically by optimizing a regression objective. A novel pushdown automaton extension was also presented for capturing long-distance word swapping. Our models outperformed Stanford textual entailment system on all but one of the genres on the STS task.

Acknowledgements

We gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181 and the support of the DARPA Broad Operational Language Translation (BOLT) program through IBM. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

References

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. 2001. *Introduction to Algorithms, Second Edition*. MIT Press.

I. Dagan, O. Glickman, and B. Magnini. 2005. The PASCAL recognising textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*.

Y. Dombb, O. Lipsky, B. Porat, E. Porat, and A. Tsur. 2010. The approximate swap and mismatch edit distance. *Theoretical Computer Science*, 411(43).

J. Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proceedings of ACL*.

J. Esparza and A. Kucera. 2005. Quantitative analysis of probabilistic pushdown automata: Expectations and variances. In *Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science*.

K. Knight and Y. Al-Onaizan. 1998. Translation with finite-state devices. In *Proceedings of AMTA*.

S. Kumar and W. Byrne. 2003. A weighted finite state transducer implementation of the alignment template model for statistical machine translation. In *Proceedings of HLT/NAACL*.

D. C. Liu and J. Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Math. Programming*, 45:503–528.

G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. 1990. WordNet: an on-line lexical database. *International Journal of Lexicography*, 3(4).

S. Pado, D. Cer, M. Galley, D. Jurafsky, and C. Manning. 2009. Measuring machine translation quality as semantic equivalence: A metric based on entailment features. *Machine Translation*, 23:181–193.

E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. C. Carrasco. 2005. Probabilistic finite-state machines part I. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1013–1025.

M. Wang and C. Manning. 2012. SPEDE: Probabilistic edit distance metrics for MT evaluation. In *Proceedings of WMT*.

D. Wu, 2010. *CRC Handbook of Natural Language Processing*, chapter Alignment, pages 367–408. CRC Press.