ETS: Discriminative Edit Models for Paraphrase Scoring

Michael Heilman and **Nitin Madnani**
Educational Testing Service
660 Rosedale Road
Princeton, NJ 08541, USA
{mheilman,nmadnani}@ets.org

Abstract

Many problems in natural language processing can be viewed as variations of the task of measuring the semantic textual similarity between short texts. However, many systems that address these tasks focus on a single task and may or may not generalize well. In this work, we extend an existing machine translation metric, TERp (Snover et al., 2009a), by adding support for more detailed feature types and by implementing a discriminative learning algorithm. These additions facilitate applications of our system, called PERP, to similarity tasks other than machine translation evaluation, such as paraphrase recognition. In the SemEval 2012 Semantic Textual Similarity task, PERP performed competitively, particularly at the two surprise subtasks revealed shortly before the submission deadline.

1 Introduction

Techniques for measuring the similarity of two sentences have various potential applications: automated short answer scoring (Nielsen et al., 2008; Leacock and Chodorow, 2003), question answering (Wang et al., 2007), machine translation evaluation (Przybocki et al., 2009; Snover et al., 2009a), etc.

An important aspect of this problem is that similarity is not binary. Sentences can be very semantically similar, such that they might be called paraphrases of each other. They might be completely different. Or, they might be somewhere in between. Indeed, it is arguable that all sentence pairs (except exact duplicates) lie somewhere on a continuum of similarity. Therefore, it is desirable to develop methods that model sentence pair similarity on a continuous, or at least ordinal, scale.

In this paper, we describe a system for measuring the semantic similarity of pairs of short texts. As a starting point, we use the Translation Error Rate Plus (Snover et al., 2009a), or TERp, system, which was specifically developed for machine translation evaluation. TERp takes two sentences as input, finds a set of weighted edits that convert one into the other with low overall weight, and then produces a length-normalized score. TERp also has a greedy, heuristic learning algorithm for inducing weights from labeled sentence pairs in order to increase correlations with human similarity scores.

Some features of the original TERp make adaptation to other semantic similarity tasks difficult, including its largely one-to-one mapping of features to edits and its heuristic, greedy learning algorithm. For example, there is a single feature for lexical substitution, even though it is clear that different types of substitutions have different effects on similarity (e.g., substituting "43.6" with "17" versus substituting "a" for "an"). In addition, the heuristic learning algorithm, which involves perturbing the weight vector by small amounts as in grid search, seems unscalable to larger sets of overlapping features.

Therefore, here, we use TERp's inference algorithms that find low cost edit sequences but use a discriminative learning algorithm based on the Perceptron (Rosenblatt, 1958; Collins, 2002) to estimate edit cost parameters, along with an expanded feature set for broader coverage of the phenomena that are relevant to sentence-to-sentence similarity. We

*First Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 529–535,
Montréal, Canada, June 7-8, 2012.

refer to this new approach as Paraphrase Edit Rate with the Perceptron (PERP).

In addition to describing PERP, we discuss how it was applied for the SemEval 2012 Semantic Textual Similarity (STS) task.

2 Problem Definition

In this work, our goal is to create a system that can take as input two sentences (or short texts) x_1 and x_2 and produce as output a prediction $\hat{y}$ for how similar they are. Here, we use the 0 to 5 ordinal scale from the STS task, where increasing values indicate greater semantic similarity.

The STS task data includes five subtasks with text pairs from different sources: the Microsoft Research Paraphrase Corpus (Dolan et al., 2004) (MSRpar), The Microsoft Research Video corpus (Chen and Dolan, 2011) (MSRvid), statistical machine translation output of parliament proceedings (Koehn, 2005) (SMT-eur). For each of these sources, approximately 750 sentence pairs $\mathbf{x_1}$ and $\mathbf{x_2}$ and gold standard similarity values $\mathbf{y}$ were provided for training and development.

In addition, there were two surprise data sources revealed shortly before the submission deadline: pairs of sentences from Ontonotes (Pradhan and Xue, 2009) and Wordnet (Fellbaum, 1998) (OnWN), and machine translations of sentences from news conversations (SMT-news). For all five sources, the held-out test set contained several hundred text pairs. See the task description (Agirre et al., 2012) for additional details.

3 TER, TERp, and PERP

In this section, we briefly describe the TER and TERp machine translation metrics, and how the PERP system extends them in order to better model semantic textual similarity.

TER (Snover et al., 2006) uses a greedy search algorithm to find a set of edits to convert one of the paired input sentences into the other. We can view this set of edits as an alignment a between the two input sentences x_1 and x_2, and when two words in x_1 and x_2, respectively, are part of an edit operation, we say that those words are aligned.[1] Unlike traditional edit distance measures, TER allow for shifts—that is, edits that change the positions of words or phrases in the input sentence x_1. Essentially, TER searches among a set of possible shifts of the phrases in x_1 to find a set of shifts that result in the least cost alignment, using edits of other types, between x_2 and the shifted version of x_1. TER allows one to specify costs for different edit types, but it does not include a method for learning those costs from data.

TERp (Snover et al., 2009b; Snover et al., 2009a) extends TER in two key ways. First, TERp includes new types of edits, including edits for substitution of synonyms, word stems, and phrasal paraphrases extracted from a pivot-based paraphrase table (§3.1). Second, it includes a heuristic learning algorithm for inferring cost parameters from labeled data. TERp includes 8 types of edits: match (M), insertion (I), deletion (D), substitution (S), stemming (T), synonymy (Y), shift (Sh), and phrase substitution (P). The edits are mutually exclusive, such that synonymy edits do not count as substitutions, for example. TERp has 11 total parameters, with a single parameter for each edit except for phrase substition, which has four.

PERP has a general framework similar to that of TERp. It extends TERp, however, by including additional edit parameters, and by using a discriminative learning algorithm (see §5) to learn parameters rather than the heuristic technique used by TERp. Thus, PERP uses the same greedy algorithm as TERp for finding the optimal sets of edits given the cost parameters, but it allows the cost for an individual edit to depend on multiple, overlapping features of that edit. For example, costs for substitution edits depend on whether the aligned words are pronouns, whether the aligned words represent numbers, the lengths of the aligned words, etc. See §4 for the full list of features in PERP.

An alignment from the MSRpar portion of the STS training data is illustrated in Figure 1.

3.1 Phrasal Paraphrases

PERP uses probabilistic phrasal substitutions to align phrases in the hypothesis with phrases in the

[1] For machine translation evaluation with TERp and PERP, x_1 is a system's hypothesis and x_2 is a reference translation. For all STS subtasks, we assigned sentences in the first and second columns of the input files to x_2 and x_1, respectively, so that the hypotheses and references in the SMT-eur subtask would be assigned appropriately.

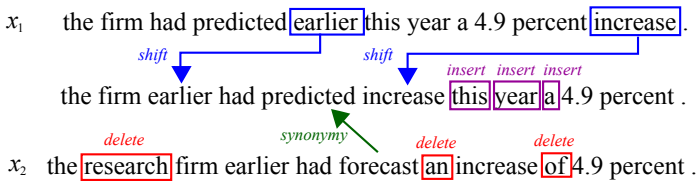

Figure 1: An example of a PERP alignment for a sentence pair from the Microsoft Research Paraphrase Corpus. The search algorithm first performs shifts on x_1 and then performs other edits on x_2. The zero cost edits that match individual words are not shown.

reference. It does so by looking up—in a pre-computed phrase table—paraphrases of phrases in the reference and using its associated edit cost as the cost of performing a match against the hypothesis. The paraphrase table used in PERP was identical to the one used by Snover et al. (2009a). It was extracted using the pivot-based method as described by Bannard and Callison-Burch (2005) with several additional filtering mechanisms to increase the precision of the extracted pairs. The pivot-based method utilizes the inherent monolingual semantic knowledge from bilingual corpora: we first identify phrasal correspondences between English and a given foreign language F, then map from English to English by following translation units from English to the other language and back. For example, if the two English phrases $e1$ and $e2$ both correspond to the same foreign phrase f, then they may be considered to be paraphrases of each other with the following probability:

$$p(e1|e2) \approx p(e1|f)p(f|e2)$$

If there are several pivot phrases that link the two English phrases, then they are all used in computing the probability:

$$p(e1|e2) \approx \sum_{f'} p(e1|f')p(f'|e2)$$

We used the same phrasal paraphrase database as in TERp (Snover et al., 2009a), which was extracted from an Arabic-English newswire bitext containing a million sentences. A few examples of the paraphrase pairs used in the MSRpar portion of the STS training data are shown below:

(*commission* → *panel*)
(*the spying* → *espionage*)
(*suffered* → *underwent*)
(*room to* → *space for*)
(*per cent* → *percent*)

4 Features

As discussed in §3, PERP expands on TERp's original features in order to better model semantic textual similarity.

PERP models a pair of sentences x_1 and x_2 using a feature function $\mathbf{f}(a)$ that extracts a vector of real-valued features from an alignment a between x_1 and x_2. This alignment is found with TERp's inference algorithm and consists of a set of edits of various types along with information about the words on which those edits operate. For example, the alignment might contain an edit with the information, "The token 'the' in x_1 was substituted for the token 'an' in x_2." This edit would increment the features in $\mathbf{f}(a)$ for the number of substitutions and the number of substitutions of stopwords, along with other relevant substitution features.

The set of features encoded in $\mathbf{f}(a)$ are described in Table 1.[2] It includes general features that always fire for edits of a particular type (e.g., the "Substitution" feature) as well as specific features that fire only in specific situations (e.g., the "Sub-Pronoun-Both" edit, which fires only when one pronoun is substituted for another).

The function $\mathbf{f}(a)$ is normalized for sentence

[2]All words were converted to lower-case. Word frequencies were calculated from the NYT stories in the fifth edition of the English Gigaword corpus. The stories were tokenized using NLTK and words occurring fewer than 100 times were excluded. Words occurring at least 100 times constituted the vocabulary used for computing the OOV features. The OOV and frequency features only fired for words that consisted only of letters, and the frequency features did not fire for OOV words. The set of negation words including the following: "no", "not", "never", and "n't". The stopword list contained 158 common words and punctuation symbols.

Edits	Feature Name	Description
-	Intercept	Always 1 (and not normalized by text lengths)
T	Stemming	The number of times that two words with the same stem, according to the Porter (1980) stemmer, were aligned.
Y	Synonymy	The number of times that a pair of synonyms, according to WordNet (Fellbaum, 1998), were aligned.
Sh	Shift	The number of shifts.
P	Paraphrase1	The number of phrasal paraphrasing operations.
P	Paraphrase2	The sum of $q \log_{10}(p)$, where p is the probability in the pivot-based paraphrase table for a paraphrase edit and q is the number of edits for that paraphrase edit. See Snover et al. (2009a) for further explanation.
P	Paraphrase3	The sum of pq, where p and q are as above.
P	Paraphrase4	The sum of q, where q is as above.
I	Insertion	The number of insertions.
D	Deletion	The number of deletions.
I, D	Insert-Delete-LogFreq	The sum of $\log_{10}$ freq(w) over all insertions and deletions, where w is the word being inserted or deleted and freq(w) is the relative frequency of w.
I, D	Insert-Delete-LogWordLen	The sum of $\log_{10}$ length(w) over all insertions and deletions, where w is the word being inserted or deleted.
I, D	Insert-Delete-X	The number of insertions and deletions of X in alignment, where X is: (a) punctuation, (b) numbers, (c) personal pronouns, (d) negation words, (e) stop words, or (f) out-of-vocabulary (OOV) words (6 features in all).
S	Substitution	The number of substitutions.
S	Sub-X-Both	The number of substitutions where both words are: (a) punctuation, (b) numbers, (c) personal pronouns, (d) negation words, (e) stop words, or (f) OOV words (6 features in all).
S	Sub-X-1only	The number of substitutions where only one word is: (a) punctuation, (b) a number, (c) a personal pronoun, (d) a negation word, (e) a stop word, or (f) an OOV word (6 features in all).
S	Sub-LogFreq-Diff	The sum of $\lvert \log_{10}$ freq$(w_1) - \log_{10}$ freq$(w_2)\rvert$ over all substitutions.
S	Sub-Contain	The number of substitutions where both words have more than 5 characters and one is a proper substring of the other.
S	Sub-Diff-By-NonWord	The number of substitutions where the words differ only by non-alphanumeric characters.
S	Sub-Small-LevDist	The number of substitutions where both words have more than 5 characters and the Levenshtein distance between them is 1.
S	Sub-Norm-LevDist	The sum of the following over all substitutions: the Levenshtein distance between the words normalized by the length of the longer word.

Table 1: The set of features in PERP. The first column lists which edits for which each feature is relevant.

lengths by dividing all the values in Table 1 by the sum of the number of words in x_1 and x_2, except for the intercept feature that models the base similarity value in the training data and always has value 1.

There are 36 features and corresponding parameters in all, compared to 11 for TERp.

It is worth pointing out that while the mutual exclusivity between most of the original TERp edits is preserved, PERP does have shared features between insert and delete edits (e.g., "Insert-Delete-Number"), and could in principle share features between substitution, stemming, and synonymy edits.

5 Learning

Given a training set consisting of paired sentences $\mathbf{x_1}$ and $\mathbf{x_2}$ and gold standard semantic similarity ratings $\mathbf{y}$, PERP uses Algorithm 1 to induce a good set

Algorithm 1 $\texttt{learn}(\mathbf{w}, T, \alpha, \mathbf{x_1}, \mathbf{x_2}, \mathbf{y})$:
An Averaged Perceptron algorithm for learning edit cost parameters. T is the number of iterations through the dataset. α is a learning rate. $\mathbf{x_1}$ and $\mathbf{x_2}$ are paired lists of sentences, and $\mathbf{y}$ is a list of similarities that correspond to those sentence pairs.

$\mathbf{w}_{sum} = \mathbf{0}$
for $t = 1, 2, \ldots, T$ **do**
  $\mathbf{x_1}, \mathbf{x_2}, \mathbf{y} = \text{shuffle}(\mathbf{x_1}, \mathbf{x_2}, \mathbf{y})$
  for $i = 1, 2, \ldots, |\mathbf{y}|$ **do**
    $a = \text{TERpAlign}(\mathbf{w}, x_{1i}, x_{2i})$
    $\hat{y} = \mathbf{w} \cdot \mathbf{f}(a)$
    $\mathbf{w} = \mathbf{w} + \alpha(y_i - \hat{y})\mathbf{f}(a)$
    $\mathbf{w} = \text{applyShiftConstraint}(\mathbf{w})$
    $\mathbf{w}_{sum} = \mathbf{w}_{sum} + \mathbf{w}$
  end for
end for
return $\frac{\mathbf{w}_{sum}}{T|\mathbf{y}|}$

of cost parameters for its various features.[3] The algorithm is a fairly straightforward application of the Perceptron algorithm described by Collins (2002).[4] The only notable difference is that the algorithm constrains PERP's shift parameter to be at least 0.01 in the step labeled "applyShiftConstraint." We found that TERp's inference algorithm would fail if the shift cost reached zero.[5] In our experiments, we initialized all weights to 0, except for the following: the "Substitution," "Insertion," and "Deletion" weights were initialized to 1.0, and the "Shift" weight was initialized to 0.1. Following Collins (2002), the algorithm returns an averaged version of the weights, though this did not appear to substantially impact performance.

[3]The "shuffle" step shuffles the lists of sentence pairs and scores together such that their orderings are randomized but that they stay aligned with each other.

[4]There are a few hyperparameters in the learning algorithms. For our experiments, we set the number of iterations through the training data T to 200. We set the learning rate α to 0.01 to avoid large oscillations in the parameters. We did not systematically tune the hyperparameters. Other values might lead to better performance.

[5]With zero cost shifts, TERp would enter a loop and eventually exceed the amount of available memory. We also set the same minimum cost of 0.01 for shifts in our experiments with the original TERp.

6 Experiments

In this section, we report results for the STS shared task. For a full description of the task, see Agirre et al. (2012).

The task consisted of three known subtasks (MSRpar, MSRvid, and SMT-eur) and two surprise subtasks (On-WN, SMT-news). For the known subtasks, we trained models with task-specific data only. For the On-WN subtask, we used the model trained for MSRpar. For SMT-news, we used the model trained for SMT-eur.

Our submissions to the task included results from two variations, one using the full system (PERPphrases) and one with the paraphrase substitution edits disabled (PERP), in order to isolate the effect of including phrasal paraphrases. In our original submission, the PERPphrases system included a minor bug that affected the calculation of the phrasal paraphrasing features. Here, we report both the original results and a corrected version ("PERPphrases (fix)"), though the correction only minimally affected performance. We also tested two variations of the original TERp system: one with the weights set as reported by Snover et al. (2009a) ("TERp (default)"), and one tuned in the same task-specific manner as PERP ("TERp (tuned)"). We multiplied TERp's predictions by -1 since it produces costs rather than similarities.

The results, in terms of Pearson correlations with test set gold standard scores, are shown in Table 2. In addition to correlations for each subtask, we include the three aggregated measures used for the task. The "ALL" measure is the Pearson correlations on the concatenation of all the data for all five subtasks. It was the original measured used to aggregate the results for the different subtasks. The second aggregated measure is the "Allnrm" measure, which we view as an oracle because it uses the gold standard similarity values from the test set to adjust system predictions. The final aggregate measure is the mean of the correlations for the subtasks, weighted by the number of examples in each subtask's test set ("Mean"). See Agirre et al. (2012) for a full description of the metrics.

For comparison, the table also includes the results from the top-ranked submission according to the "ALL" measure, the results for the word-overlap

	Aggregated Measures			Subtask Measures				
	ALL	**ALLnrm**	**Mean**	**MSRpar**	**MSRvid**	**SMT-eur**	**On-WN**	**SMT-news**
UKP (top-ranked)	**.8239**	**.8579**	**.6773**	**.6830**	**.8739**	**.5280**	.6641	.4937
PERPphrases (fix) †	.7837	—	.6405	.6410	.7209	.4852	**.7127**	**.5312**
PERPphrases	.7834	.8089	.6399	.6397	.7200	.4850	.7124	**.5312**
PERP	.7808	.8064	.6305	.6211	.7210	.4722	.7080	.5149
TERp (tuned) †	.5558	—	.5582	.5400	.6099	.4967	.5862	.5135
TERp (default)	.4477	.7291	.5253	.5049	.5217	.4748	.6169	.4566
baseline	.3110	.6732	.4356	.4334	.2996	.4542	.5864	.3908
mean of submissions	.5864	.7773	.5286	.4894	.7049	.3958	.5557	.3731

Table 2: Pearson correlations between predictions about the test data and gold standard scores. "†" marks experiments that were not parts of the official SemEval task 6 evaluation. The highest correlation in each column is given in bold. ALLnrm results are not included for all runs because we did not have an implementation of that measure.

baseline from the organizers (Agirre et al., 2012), and the means across all 88 submissions (not including the baseline).

Table 3 shows the rankings in the official results of the PERPphrases submission, for each subtask and overall, along with Pearson correlations from PERP and the best submission for each subtask.

Aggregated Measure	**Rank**	ρ	$\rho_{\mathbf{best}}$
ALL	6	.7834	.8239
ALLnrm	27	.8089	.8635
Mean	7	.6399	.6773
Subtask Measure	**Rank**	ρ	$\rho_{\mathbf{best}}$
MSRpar	8	.6397	.7343
MSRvid	52	.7200	.8803
SMT-eur	21	.4850	.5666
On-WN	2	.7124	.7273
SMT-news	4	.5312	.6085

Table 3: The ranking and correlation (ρ) obtained by PERPphrases for each of the five datasets as well for all datasets combined. The STS task had a total of 88 submissions. ρ_{best} shows the correlation for the best submission, across all submissions, for each dataset.

7 Conclusion

From the results in §6, PERP appears to be competitive at measuring semantic textual similarity. It performed particularly well on the surprise subtasks, indicating that it generalizes well to new data. Finally, with the exception of the SMT-eur machine translation evaluation subtask, PERP outperformed the TERp system for all of the STS subtasks.

Acknowledgments

We would like to thank the organizers of SemEval and the Semantic Textual Similarity task. We would also like to thank Matt Snover for making the original TERp code available.

References

E. Agirre, D. Cer, M. Diab, and A. Gonzalez-Agirre. 2012. SemEval-2012 task 6: A pilot on semantic textual similarity. In *Proc. of the 6th International Workshop on Semantic Evaluation (SemEval 2012), in conjunction with the First Joint Conference on Lexical and Computational Semantics (*SEM 2012)*.

C. Bannard and C. Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proc. of ACL*, pages 597–604.

D. Chen and W. B. Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proc. of ACL*, pages 190–200.

M. Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with the perceptron algorithm. In *Proc. of EMNLP*.

W. Dolan, C. Quirk, and C. Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proc. of COLING*, pages 350–356, Geneva, Switzerland.

C. Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

P. Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proc. of Machine Translation Summit*.

C. Leacock and M. Chodorow. 2003. c-rater: Scoring of short-answer questions. *Computers and the Humanities*, 37.

R. D. Nielsen, W. Ward, and J. H. Martin. 2008. Classification errors in a domain-independent assessment system. In *Proc. of the Third Workshop on Innovative Use of Natural Language Processing for Building Educational Applications*.

M. F. Porter. 1980. An algorithm for suffix stripping. *Program*, 3(14):130–137.

S. S. Pradhan and N. Xue. 2009. OntoNotes: The 90% solution. In *Proc. of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Tutorial Abstracts*, pages 11–12.

M. A. Przybocki, K. Peterson, S. Bronsart, and G. A. Sanders. 2009. The NIST 2008 metrics for machine translation challenge - overview, methodology, metrics, and results. *Machine Translation*, 23(2-3):71–103.

F. Rosenblatt. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65.

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A study of Translation Edit Rate with targeted human annotation. In *Proc. of the Conference of the Association for Machine Translation in the Americas (AMTA)*.

M. Snover, N. Madnani, B. Dorr, and R. Schwartz. 2009a. Fluency, adequacy, or HTER? Exploring different human judgments with a tunable MT metric. In *Proc. of the Fourth Workshop on Statistical Machine Translation at the 12th Meeting of the European Chapter of the Association for Computational Linguistics (EACL-2009)*, March.

M. Snover, N. Madnani, B. Dorr, and R. Schwartz. 2009b. TER-Plus: Paraphrase, semantic, and alignment enhancements to Translation Edit Rate. *Machine Translation*, 23(2–3):117–127.

M. Wang, N. A. Smith, and T. Mitamura. 2007. What is the Jeopardy model? A quasi-synchronous grammar for QA. In *Proc. of of EMNLP*.