CS6390 PROJECT PROPOSAL: REDIRE/GIRAF

OLIVER RICHARDSON MAKS CEGIELSKI-JOHNSON

1. Task

Our interest in this problem is two-fold; we are interested in studying both recognition and generation of paraphrases. Hence we will need to discuss our current approaches to both of these problems. 4 The paraphrase recognition task is defined as attempting to determine whether two sentences are paraphrases of each other, returning a YES/NO answer.

The paraphrase generation task is defined as being given an input sentence, and trying to generate a paraphrase from that input.

2. Architecture

Our current implementation is inspired by a paper by Malakasiotis[2]. The general structure of this implementation is as follows:

(1) First, we read all sentence pairs (s_1, s_2) from a file.

(2) Restructure each pair of sentences s_1 and s_2 into 10 new string transformations $t_{i,1}$ and $t_{i,2}$, maintaining the original order

(a) String of the tokens
(b) String of the stems.
(c) String of the POS tags.
(d) String of the soundex codes[1].
(e) String of all the noun tokens
(f) String of all the noun stems
(g) String of all the noun soundex codes.
(h) String of all the verb tokens.
(i) String of all the verb stems.
(j) String of all the verb soundex codes.

(3) For each string pair s_1 and s_2, and for each transformation of that string $t_{i,1}$ and $t_{i,2}$, we want to compute the similarity [3] $d_i(t_{i,1}, t_{i,2})$, using the metrics

(a) Levenshtein (word edit)
(b) Jaro-Winkler
(c) Manhattan & Euclidean
(d) Cosine
(e) n-gram ($n = 3$)
(f) Overlap
(g) Dice coefficient
(h) Jaccard coefficient

(4) Computing the similarities between each transformation of the two strings results in 90 features.

(5) Next, for each pair of sentences s_1 and s_2, if s_1 is longer than s_2 we compute the substrings of s_1 which have the same length as s_2. For each substring s_1' generated, we compute all of the similarities between s_1' and s_2. We try to find which substring results in the max average similarity over all the similarity metrics, namely $s_1'^*$. Then for $s_1'^*$ we compute each of the similarities between it and s_2, as well as the average of all of them, resulting in 10 features. We do this for the string transformations which result in tokens, stems, POS tags,

and soundex codes. This results in an additional 40 features per sentence pair (total of 130 features so far). Note that we can just swap the two strings in the discussion above if s_2 is larger than s_1.

(6) Finally, we add three additional features to the current 130. We add a boolean feature for whether s_1 contains a negation, and a feature for whether s_2 contains negation. Then finally, we add a feature for the length ratio, defined as $\frac{\min(|s_1|,|s_2|)}{\max(|s_1|,|s_2|)}$.

(7) Optionally, we can add one more set of features. We can compute the dependency parses for both sentences s_1 and s_2. Using the dependency parses, we can calculate the following

$$R_1 = \frac{|\text{common dependencies}|}{|\text{dependencies in } s_1|} \qquad R_2 = \frac{|\text{common dependencies}|}{|\text{dependencies in } s_2|}$$

which we can both use as features, as well as the harmonic mean $\frac{2R_1R_2}{R_1+R_2}$.

3. Experiments

Having defined how to get our similarity-based features, we can start experimenting with different combinations. First, let us define a baseline, (BASE), for which we will use the Levenshtein distance and a simple linear classifier. We can see the results of this baseline in Table 1.

4. Results

Method	**Accuracy**	**Precision**	**Recall**	**F-Score**
BASE	0.64811	0.68442	0.87358	0.76752
SIM	0.66492	0.72814	0.79163	0.75856
SIM + DEP	0.66492	0.73224	0.78204	0.75632

Table 1. REDIRE results

Method	**Accuracy**	**Precision**	**Recall**	**F-Score**
BASE	0.6904	0.7242	0.8631	0.7876
INIT	0.7519	0.7851	0.8631	0.8223
INIT + WN	0.7548	0.7891	0.8614	0.8237
INIT + WN + DEP	0.7935	0.7891	0.8675	0.8288

Table 2. Malakasiotis results

5. Conclusions

Clearly, following the Malakasiotis paper has not given us comparable results, netting a 7% lower F-score using just similarity metrics. The next step would be to determine what is causing the much lower score. However, we are still happy with the progress we currently have. Clearly from Table 2 we can see that adding WordNet (WN) did not boost performance significantly, so we don't necessarily need to focus our attention on implementing this functionality. However, both REDIRE and Malakasiotis for SIM and INIT respectively only use naïve word similarity metrics.

6. Contributions

References

[1]

[2] Prodromos Malakasiotis. Paraphrase recognition using machine learning to combine similarity measures. In *Proceedings of the ACL-IJCNLP 2009 Student Research Workshop*, pages 27–35. Association for Computational Linguistics, 2009.

[3] Prodromos Malakasiotis and Ion Androutsopoulos. Learning textual entailment using svms and string similarity measures. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 42–47. Association for Computational Linguistics, 2007.