AUEB at TAC 2008

Dimitrios Galanis and Prodromos Malakasiotis
Department of Informatics
Athens University of Economics and Business
Patission 76, GR-104 34 Athens, Greece

Abstract

This paper describes AUEB's participation in TAC 2008. Specifically, we participated in the summarization and textual entailment recognition tracks. For the former we trained a Support Vector Regression model that is used to rank the summary's candidate sentences; and for the latter we used a Maximum Entropy classifier along with string similarity measures applied to several abstractions of the original texts.

1 Introduction

Over the past years, several challenges and workshops concerning subareas of Natural Language Processing (e.g. question answering, textual entailment recognition, summarization etc.) have been organized. This year the National Institute of Standards and Technology (NIST) organized the Text Analysis Conference (TAC) 2008, which is a series of workshops providing the infrastructure for large-scale evaluation of NLP technology. The conference consists of three main tracks, namely question answering, summarization, and textual entailment recognition. Summarization is further split into two tasks: update summarization and opinion summarization, which is a pilot task. In this paper we present AUEB's's[1] participation in the update summarization and textual entailment recognition tracks. In section 2 we present our submission for the update summarization track, while section 3 describes our participation in the recognizing textual entailment track.

2 Update Summarization

Query-Focused Summarization (QFS) is the task of synthesizing a coherent well structured answer to a complex question from a set of documents [Dang, 2005, 2006]. Today's QFS systems produce summaries by extracting the most salient sentences of the original documents.

Until recently, the salience of each sentence was usually calculated using a weighted linear combination of features, where the weights were either assigned by experience or by a trial and error process. Recently, Support Vector Regression (SVR) has been used to combine these features yielding very satisfactory results in DUC 2007. Li et al. [2007] trained their SVR model on past DUC data documents. In particular, for every sentence of the documents one training vector was constructed by calculating: a) some predetermined features and b) a label (a score) which indicates the similarity of the sentence to sentences in the model summaries that were constructed by the DUC judges. The trained model is used to determine the relevance of a sentence to a given complex query. In DUC 2007, Li et al. [2007] 's system ranked 5th in ROUGE-2 and ROUGE-SU4 and 15th in responsiveness

Schilder and Ravikumar [2008] adopt a very similar approach with simple features and a score which is calculated as the word overlap between the sentence that was extracted from a document and the sentences in DUC model summaries. Their results are very satisfactory as they ranked in 6th and 2nd in ROUGE-2 in DUC 2007 and 2006 respectively.

We propose a different way to assign the score to each training example. We use a combination of the ROUGE-2 and ROUGE-SU4 score [Lin, 2004], because these scores have strong correlation with the content responsiveness score which is assigned by the human judges and measures the information coverage of the summaries. In this way we believe that our system will select more relevant sentences. We also experiment with different sizes of training sets.

2.1 SVR Training

In our system the following features were used:

- Sentence position $SP(c)$:

$$SP(c) = \frac{position(c)}{length(c)}$$

  where c is a sentence, $position(c)$ is the position of c in its document and length(c) is the length of c.

[1] `http://pages.cs.aueb.gr/nlp`

- Named entities $NE(c)$:

$$NE(c) = \frac{n(c)}{length(c)}$$

  where n(c) is the number of named entities in c.

- Levenstein distance LD(c,q): The Levenstein Distance between the query (q) and the sentence (c)

- Word overlap WO(c, q): The word overlap between the query (q) and the sentence (c). Stop words and duplicate words have been removed.

- Content word frequency $CF(c)$ and document frequency $DF(c)$ as defined in [Schilder and Ravikumar, 2008].

In order to train the SVR, the DUC 2006 documents were used. All sentences of all the documents were extracted and a training vector was constructed for each one of them, using the aforementioned features. The score which was assigned to each vector was calculated as the average of the ROUGE-2 and ROUGE-SU4 of the sentence with the corresponding four model summaries.

2.2 Summarizer

To evaluate our system we used it to construct summaries of the DUC 2007 documents. In particular, the SVR was used to assign relevance scores to all sentences of the DUC 2007 data, and according to the scores a sorted list of sentences was created for each query. Starting from the sentence with the highest score, the system added to the summary every sentence whose similarilty to the sentences already contained in the summary did not exceed a threshold. The similarity was measured using cosine similarity and the threshold was determined by experiments on DUC 2006 corpus.

Furthermore, before a sentence was added to the summary it was simplified through simple heuristics. Specifically, the strings "However ," , "In fact" , "At this point ," , "As a matter of fact ," , ", however ," and , "also ," were deleted and the same was done for some temporal phrases like "here today" and "Today". Finally, the summaries were truncated to form summaries of 250 words, which is the upper limit in DUC 2007.

The system achieved 0.113 in ROUGE-2 and 0.165 in ROUGE-SU4, being 5th in both categories, having slightly higher scores from the other SVR based systems [Li et al., 2007; Schilder and Ravikumar, 2008] that have been evaluated on DUC 2007 data.

Also, we experimented with different sizes of the training set. In these experiments, we didn't use all the trimming heuristics, which is why the ROUGE scores are slightly worse that those reported in the previous section. The results are presented in table 1 which shows that the summarizer achieves the best results when it is trained with all of the available training examples. In addition, the results show that sentence simplification affects significantly the ROUGE scores.

training vectors	ROUGE-2	ROUGE-SU4
35000	0.10916 (6th)	0.15959 (10th)
22000	0.10769 (9th)	0.15892 (10th)
11000	0.10807 (8th)	0.15835 (12th)
1000	0.10077 (16th)	0.15017 (18th)
10	0.10329 (13th)	0.15313 (16th)
2	0.06508 (30th)	0.11731 (31th)

Table 1: Our system's ROUGE scores for different training sizes. The system is trained on DUC 2006 data and it tested on DUC 2007 data.

2.3 Official results and discussion

The UPDATE TASK in TAC 2008 was to produce summaries for 48 complex queries provided by the organizers. For each query, two sets of documents, namely A and B, were also provided and the task was to produce two summaries, each one containing maximum of 100 words. The first summary should summarize the documents contained in set A, and the second summary the documents contained in B given that the reader has already read the A.

The summaries for the set A were produced using the summarizer that was described in the previous section. For the the summaries of set B, we used the same algorithm, but we filtered out the sentences having high similarity to any of the sentences of set A. We used the same cosine similarity and threshold as before.

Each team was able to submit up to three runs of its system, however, we submitted only one run. The system's results (team id 2) in TAC 2008 are presented in table 2.[2] The system was trained with all sentences of DUC 2006.

evaluation	rank	score
ROUGE-2	4th / 72 runs	0.09623
ROUGE-SU4	4th / 72 runs	0.13435
BE	19th / 72 runs	0.05199
modified pyramid	17th / 58 runs	0.28000
overall responsiveness	18th / 58 runs	2.38500
linguistic quality	31th / 58 runs	2.35400

Table 2: System's results in TAC 2008

Given that the system does not employ sophisticated sentence simplification algorithms its rankings

[2]In human evaluations only two runs for each team were evaluated.

especially in the automatic evaluations, are very satisfactory. In human evaluations, and more specifically in the linguistic quality score the system, as expected, did not achieve a good ranking because it does not employ algorithms for ordering and rewriting of the selected sentences. We believe that the low linguistic quality score affects also the overall responsiveness score of the system, something that is also observed in [Conroy and Dang, 2008].

In future work we will try to use other methods to train the SVR.For example, in [Giannakopoulos et al., 2008, to appear] an alternative method of ROUGE is presented for content evaluation which achieves high correlation with human judges. In addition, we want to improve the linguistic quality of our summaries by employing rewriting and sentence ordering algorithms.

3 Recognizing textual entailment

Textual Entailment is of significant importance in many natural language processing areas, such as question answering, information extraction, information retrieval, and multi-document summarization. In the TAC Recognizing Textual Entailment Challenge (RTE), it is defined as the task of deciding whether or not the meaning of a *hypothesis* text (H) can be inferred from the meaning of another text (T).[3] For instance, the following is a correct entailment pair:

T: A bus collision with a truck in Uganda has resulted in at least 30 fatalities and has left a further 21 injured.

H: 30 die in a bus collision in Uganda.

If the meaning of H cannot be inferred from the meaning of T, either T contradicts H (contradiction pair) or the truth of H cannot be judged on the basis of T (unknown pair). The first pair bellow is "contradiction pair", whereas the second one is an "unknown pair":

T: Gastrointestinal bleeding can happen as an adverse effect of non-steroidal anti-inflammatory drugs such as aspirin or ibuprofen.

H: Aspirin prevents gastrointestinal bleeding.

T: Blue Mountain Lumber said today it may have to relocate a $30 million project offshore in the wake of an Environment Court decision that blocked it from a planned development site on the Coromandel.

H: Blue Mountain Lumber will locate a development site on the Coromandel.

So, this year's challenge was a three-way classification task, but the original two-way task was also preserved. Each team could submit up to three runs per task (three-way or two-way).

[3]See `http://www.pascal-network.org/`.

In the following section, we describe our participation in the TAC 2008 RTE track. We used a supervised machine learning algorithm with string similarity measures as features. We also employed feature selection techniques and used WordNet [Fellbaum, 1998] at a preprocessing step.

3.1 System overview

Our system uses a Maximum Entropy (ME) [Jaynes, 1957; Good, 1963] classifier[4] to distinguish between the three different categories (namely entailment, contradiction, and unknown) that a T–H pair can be classified in. The classifier is trained with vectors having as features string similarity measures, under the assumption that similarities at various shallow abstractions of the input (e.g., the original sentences, the stems of their words, their POS tags) can be used to recognize textual entailment reasonably well. This approach attempts to improve our previous system [Malakasiotis and Androutsopoulos, 2007] that participated in the 3rd RTE challenge [Giampiccolo et al., 2007]. We now try to exploit WordNet as well, as already mentioned, and we also use more advanced feature selection techniques, to be discussed later on.

We employ 9 string similarity measures that are applied to the following 10 pairs of strings, which correspond to 10 different levels of abstraction of T and H. These pairs are:

pair 1: two strings consisting of the *original tokens* of T and H, respectively, with the original order of the tokens maintained;[5]

pair 2: as in the previous case, but now the tokens are replaced by their *stems*;

pair 3: as in the previous case, but now the tokens are replaced by their *part-of-speech* (POS) tags;

pair 4: as in the previous case, but now the tokens are replaced by their *soundex codes*;[6]

pair 5: two strings consisting of only the *nouns* of T and H, as identified by a POS-tagger, with the original order of the nouns maintained;

pair 6: as in the previous case, but now with *nouns replaced by their stems*;

pair 7: as in the previous case, but now with *nouns replaced by their soundex codes*;

pair 8: two strings consisting of only the *verbs* of T and H, as identified by a POS-tagger, with the original order of the verbs maintained;

pair 9: as in the previous case, but now with *verbs replaced by their stems*;

[4]We use Stanford University's implementation; see `http://nlp.stanford.edu/`.

[5]We use Stanford University's tokenizer and POS-tagger, and our own implementation of Porter's stemmer.

[6]Soundex is an algorithm intended to map English names to alphanumeric codes, so that names with the same pronunciations receive the same codes, despite spelling differences; see `http://en.wikipedia.org/wiki/Soundex`.

pair 10: as in the previous case, but now with *verbs replaced by their soundex codes.*

A common problem in textual entailment is that T may be much longer than H, which may mislead the string similarity measures. Consider, for example, the following T–H pair where H appears almost verbatim in T, but the length difference yields low similarity.

T: <u>Anna Politkovskaya was found shot dead</u> on Saturday in a lift at her block of flats in the Russian capital, Moscow.

H: Anna Politkovskaya was murdered.

To address this problem, when we consider a pair of strings (s_1, s_2), if s_1 is longer than s_2, we also compute the nine values $f_i(s_1', s_2)$, where f_i $(1 \leq i \leq 9)$ are the string similarity measures, for every s_1' that is a substring of s_1 of the same length as s_2. We then locate the s_1' with the best average similarity to s_2, shown below as $s_1'^{*}$:

$$s_1'^{*} = \arg\max_{s_1'} \sum_{i=1}^{9} f_i(s_1', s_2)$$

and we keep the nine $f_i(s_1'^{*}, s_2)$ values and their average as 10 additional measurements. Similarly, if s_2 is longer than s_1, we keep the nine $f_i(s_1, s_2'^{*})$ values and their average. This process is applied only to pairs 1–4; hence, there is a total of 40 additional measurements in each T–H case.

The measurements discussed above provide 130 numeric features that can be used by the induced classifier.[7] To those, we add two Boolean features indicating the existence or absence of negation in T or H, respectively; negation is detected by looking for words like “not”, “won’t” etc. Finally, we add a length ratio feature, defined as $\frac{\min(L_T, L_H)}{\max(L_T, L_H)}$, where L_T and L_H are the lengths, in tokens, of T and H. Hence, there is a total of 133 available features.

3.1.1 String similarity measures

We now describe the nine string similarity measures that we use. The reader is reminded that the measures are applied to string pairs (s_1, s_2), where s_1 and s_2 correspond to the ten aforementioned abstractions of T and H, respectively.

Levenshtein distance: This is the minimum number of operations (edit distance) needed to transform one string (in our case, s_1) into the other one (s_2), where an operation is an insertion, deletion, or substitution of a single character. In pairs of strings that contain POS tags and soundex codes, we consider operations that insert, delete, or substitute entire tags, instead of characters.

[7]All feature values are normalized in $[-1, 1]$. We use our own implementation of the string similarity measures.

Jaro-Winkler distance: The Jaro-Winkler distance [Winkler, 1999] is a variation of the Jaro distance [Jaro, 1995], which we describe first. The Jaro distance d_j of s_1 and s_2 is defined as:

$$d_j(s_1, s_2) = \frac{m}{3 \cdot l_1} + \frac{m}{3 \cdot l_2} + \frac{m - t}{3 \cdot m},$$

where l_1 and l_2 are the lengths (in characters) of s_1 and s_2, respectively. The value m is the number of characters of s_1 that match characters of s_2. Two characters from s_1 and s_2, respectively, are considered to match if they are identical and the difference in their positions does not exceed $\frac{\max(l_1, l_2)}{2} - 1$. Finally, to compute t (‘transpositions’), we remove from s_1 and s_2 all characters that do not have matching characters in the other string, and we count the number of positions in the resulting two strings that do not contain the same character; t is half that number.

The Jaro-Winkler distance d_w emphasizes prefix similarity between the two strings. It is defined as:

$$d_w(s_1, s_2) = d_j(s_1, s_2) + l \cdot p \cdot [1 - d_j(s_1, s_2)],$$

where l is the length of the longest common prefix of s_1 and s_2, and p is a constant scaling factor that also controls the emphasis placed on prefix similarity. The implementation we used considers prefixes up to 6 characters long, and sets $p = 0.1$.

Again, in pairs of strings (s_1, s_2) that contain POS tags or soundex codes, we apply this measure to the corresponding lists of tags in s_1 and s_2, instead of treating s_1 and s_2 as strings of characters.

Manhattan distance: Also known as City Block distance or L_1, this is defined for any two vectors $\vec{x} = \langle x_1, \ldots, x_n \rangle$ and $\vec{y} = \langle y_1, \ldots, y_n \rangle$ in an n-dimensional vector space as:

$$L_1(\vec{x}, \vec{y}) = \sum_{i=1}^{n} |x_i - y_i|.$$

In our case, n is the number of distinct words (or POS tags or soundex codes) that occur in s_1 and s_2 (in any of the two); and x_i, y_i show how many times each one of these distinct words occurs in s_1 and s_2, respectively.

Euclidean distance: This is defined as follows:

$$L_2(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}.$$

In our case, $\vec{x}$ and $\vec{y}$ correspond to s_1 and s_2, respectively, as in the previous measure.

Cosine similarity: The definition follows:

$$\cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\| \cdot \|\vec{y}\|}.$$

In our system $\vec{x}$ and $\vec{y}$ are as above, except that they are binary, i.e., x_i and y_i are 1 or 0, depending on whether or not the corresponding word (or POS tag or soundex code) occurs in s_1 or s_2, respectively.

N-gram distance: This is the same as L_1, but instead of words we use all the (distinct) character n-grams in s_1 and s_2; we used $n = 3$.

Matching coefficient: This is $|X \cap Y|$, where X and Y are the sets of (unique) words (or tags) of s_1 and s_2, respectively; i.e., it counts how many common words s_1 and s_2 have.

Dice coefficient: This is the following quantity; in our case, X and Y are as in the previous measure.

$$\frac{2 \cdot |X \cap Y|}{|X| + |Y|}$$

Jaccard coefficient: This is defined as $\frac{|X \cap Y|}{|X \cup Y|}$; again X and Y are as in the matching coefficient.

3.1.2 WordNet preprocessing

Using only string similarity measures has the risk of missing true entailment relationships that are due to the existence of synonyms. Therefore, before we apply the similarity measures discussed earlier, we employ a preprocessing step during which every word of H that has a synonym in T is replaced by that synonym. We use WordNet [Fellbaum, 1998] to locate synonyms.

3.2 Feature selection

Larger feature sets do not necessarily lead to improved classification performance. Despite seeming useful, some features may in fact be too noisy or irrelevant, increasing the risk of overfitting the training data. Some features may also be redundant, given other features; thus, feature selection methods that consider the value of each feature on its own (e.g., information gain) may lead to suboptimal feature sets.

Finding the best subset of a set of available features is a search space problem for which several methods have been proposed [Guyon et al., 2006]. We used a wrapper approach, whereby each feature subset is evaluated according to the predictive power of a ME classifier (treated as a black box). More precisely, during feature selection we conducted 10-fold cross validation on the training data to evaluate the predictive power, measured as accuracy (i.e., correct decisions over all decisions) of each feature subset.

With large feature sets, an exhaustive search over all subsets is intractable. Instead, we experimented with forward hill-climbing and beam search [Guyon et al., 2006]. Forward hill-climbing starts with an empty feature set, to which it adds features, one at a time, by preferring to add at each step the feature that leads to the highest predictive power. Forward hill climbing has a high risk of being trapped in local maxima. Therefore, we have also experimented with forward beam search, which is similar, except that the search frontier contains the k best examined feature subsets at each time.

3.3 Official results and discussion

We submitted a total of six runs, three for the three-way classification task and three for the two-way classification task. The three runs were produced in the same way for both tasks. For the first run we trained our system using all the features described in section 3.1. In the second run, the classifier was trained with the best features selected by forward hill climbing. Finally, in the third run we used forward beam search with $k = 10$ to select the best features. The runs for the three-way task were also evaluated as two-way runs, simply by merging "contradiction" and "unknown" pairs to "no entailment" pairs.

As training data we used the development and evaluation data of the third RTE challenge. Preliminary experiments indicated that the use of additional data from other challenges (e.g., including training data from the second RTE challenge) reduces the predictive power of the classifier. This might be due to differences in the ways the datasets were constructed. Tables 3 and 4 present the results of our runs. The scores in brackets are the corresponding cross validation scores we had obtained on the training data during feature selection.

The cross validation scores were much higher, which is most probably due to differences in the manner that the two datasets we used were constructed. Moreover, we observe that feature selection can lead to similar or slightly better results, compared to using the full set of available features, which is an indication of redundancy in the complete feature set. A possible improvement could be to use a dependency parser, but we deliberately avoided using one, since we are interested in less spoken languages, for which such tools are difficult to obtain.

4 Conclusion

We presented AUEB's participation in two TAC 2008 tracks, namely summarization and RTE. For the former, we used Support Vector Regression and assigned to each training example a score equal to the average of ROUGE-2 and ROUGE-SU4. As expected, our system ranked high when the evaluation measure was either of these two scores. However, since we do not employ ordering and rewriting techniques

Two-way runs					
Run 1 (0.6425)		Run 2 (0.6831)		Run 3 (0.6875)	
Accuracy	Average precision	Accuracy	Average precision	Accuracy	Average precision
0.5660	0.5464	0.5780	0.5632	0.5660	0.5465

Table 3: RTE two-way runs.

Three-way runs					
Three-way results					
Run 1 (0.6288)		Run 2 (0.6650)		Run 3 (0.6744)	
Accuracy		Accuracy		Accuracy	
0.5460		0.5470		0.5540	
Two-way results					
Run 1		Run 2		Run 3	
Accuracy	Average precision	Accuracy	Average precision	Accuracy	Average precision
0.5800	0.5654	0.5790	0.5620	0.5840	0.5220

Table 4: RTE three-way runs.

our system does not achieve good results in human evaluations, and especially in the linguistic quality score. This also affects the overall responsiveness score of the system. We plan to improve the linguistic quality of the summaries we produce and to experiment with other content selection techniques. We hope that these enhancements will also improve the overall responsiveness of our system.

In the RTE track, we attempted to improve the system with which we participated in the 3rd RTE challenge. Therefore, apart from employing a supervised learning algorithm and string similarity measures, we also exploited WordNet and used more sophisticated feature selection techniques. We observed a large difference between the predictive power of the feature set, measured via 10-fold cross validation on the training data, and the corresponding evaluation results. This is most likely due to differences in the way the datasets we used were constructed. Moreover, the feature selection techniques lead to similar or slightly better results, which indicates possible redundancy in the complete feature set.

References

J. Conroy and H. T. Dang. Mind the gap: dangers of divorcing evaluations of summary content from linguistic quality. In *Proceedings of COLING*, 2008.

Hoa Trang Dang. Overview of DUC 2005. In *Proceedings of DUC*, 2005.

Hoa Trang Dang. Overview of DUC 2006. In *Proceedings of DUC*, 2006.

C. Fellbaum, editor. *WordNet: an electronic lexical database.* MIT Press, 1998.

D. Giampiccolo, B. Magnini, I. Dagan, and B. Dolan. The 3rd PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9, Prague, Czech Republic, 2007.

G. Giannakopoulos, V. Karkaletsis, G. Vouros, and P. Stamatopoulos. Summarization system evaluation revisited: N-gram graphs. *ACM Transactions on Speech and Language Processing*, 2008, to appear.

I. J. Good. Maximum entropy for hypothesis formulation, especially for multidimentional contigency tables. *Annals of Mathem. Statistics*, 34:911–934, 1963.

I.M. Guyon, S.R. Gunn, M. Nikravesh, and L. Zadeh, editors. *Feature Extraction, Foundations and Applications.* Springer, 2006. URL `http://eprints.ecs.soton.ac.uk/11922/`.

M.A. Jaro. Probabilistic linkage of large public health data file. *Statistics in Medicine*, 14:491–498, 1995.

E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106:620–630, 1957.

S. Li, Y. Ouyang, W. Wang, and B. Sun. Multi-document summarization using support vector regression. In *Proceedings of DUC*, 2007.

C.W. Lin. ROUGE: A package for automatic evaluation of summaries. In *Proceedings of ACL-04 Workshop: Text Summarization Branches Out*, pages 74–81, 2004.

P. Malakasiotis and I. 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, Prague, Czech Republic, 2007.

F. Schilder and K. Ravikumar. Fastsum: Fast and accurate query-based multi-document summarization. In *Proceedings of ACL*, 2008.

W.E. Winkler. The state of record linkage and current research problems. Statistical Research Report RR99/04, US Bureau of the Census, Washington, DC, 1999.