The text categorization system TEKLIS at TREC-6

Thomas Brückner

Siemens AG
Otto-Hahn-Ring 6
81730 München, Germany
thomas.brueckner@mchp.siemens.de

Abstract

This short paper documents our participation on the filtering and routing tasks of TREC-6 with the commercial filtering system TEKLIS. TEKLIS is a training-based statistical categorization system which incorporates shallow linguistic processing and fuzzy-set methods.

In the following we will present the core technology of TEKLIS, our results on the filtering and routing tasks and a discussion of the insights we gained through our participation.

1 Introduction

TEKLIS is a text categorization system aimed at a wide variety of applications like news indexing, user profile based filtering or hierarchical cataloguing of WWW sites. Since it is a true filtering system it is obviously well suited for the filtering task of TREC but it can be easily used for routing too.

As a first time participant we concentrated on getting our system to run with the TREC data and instead of making various experiments made a thorough analysis of test runs on older TREC data to see what went possibly wrong and why. We will present the results in the discussion section.

2 System Overwiew

In the first processing step of a text TEKLIS normalizes the words with a lemmatizer. The lemmatizer returns for each word one or more stems and it's lexical category. A statistical HMM Part-of-Speech Tagger resolves lexical ambiguities.

Categorizing a text in our system depends on the relevance of words for categories. The relevance of a word for various given categories is computed from a set of training texts. We define the relevance of a word w for a category c, *rlv*(w in c), through Pearsons correlation coefficent $r(w,c)$. Negative relevances are not used, since they usually worsened the results in practice.

If we categorize a new text we get for each category c a set C of relevances for each word w in text. Now we can think of the set C as a fuzzy set with membership function $\mu_C(\mathrm{w}) = rlv(\mathrm{w\ in\ c})$. For such a set C we compute it's probability with

$$\mathrm{prob}(C) := \Sigma\, \mu_C(w)\ .\ \mathrm{p}(w),$$

where $\mathrm{p}(w)$ is interpreted as $\mathrm{p}(c|w)$. When the probability for a category c, normalized over the number of words in the text, is higher than a given threshold, the system returns the category. By varying the threshold it is possible to get different recall/precision levels (higher threshold = lower recall and higher precision, lower threshold = higher recall and lower precision).

For a more detailed description of the system see [1].

3 Filtering task

Since TEKLIS is a true filtering system we only had to create an index file for the training data and changed the output to accomodate the TREC format. Otherwise we could run the system „as is" on the filtering training and test data. As TEKLIS is designed to learn from sample documents we didn't used the topics descriptions, which might be a disadvantage (see Discussion). For predicting which thresholds are best suited for the different runs (F1, F2, ASP) we made some tests on older TREC data.

Our results on the filtering task were as shown in the table below:

Topic	#relevant	F1 (run 1)	F1 (run 2)	F2 (run 1)	F2 (run 2)	ASP (run 1)	ASP (run 2)
1	51	1	2	-43	-46	0.025	0.018
3	76	0	0	-76	-76	0.000	0.003
4	80	0	0	-80	-80	0.000	0.000
5	7	-48	-68	-61	-82	0.000	0.002
6	165	54	68	-28	-2	0.150	0.169
11	174	-2	-8	-151	-124	0.023	0.056
12	292	0	0	-269	-200	0.014	0.065
23	7	0	0	-8	-8	0.000	0.000
24	42	1	1	-43	-45	0.004	0.003
44	4	0	-2	-5	-5	0.000	0.000
54	174	0	0	-174	-175	0.000	0.000
58	18	0	0	-18	-18	0.000	0.000
62	401	-19	-35	-418	-435	0.001	0.001
77	16	-2	-2	-17	-18	0.000	0.000
78	45	0	0	-45	-45	0.000	0.000
82	82	0	-2	-85	-88	0.000	0.000
94	193	-13	-35	-217	-219	0.001	0.002
95	138	0	-8	-148	-160	0.000	0.000
100	197	0	0	-197	-197	0.000	0.000
108	314	0	0	-314	-311	0.000	0.002
111	566	-40	-70	-556	-556	0.009	0.009
114	59	0	0	-59	-59	0.000	0.000
118	89	-32	-165	-154	-154	0.006	0.006
119	85	-6	-11	-85	-102	0.009	0.007
123	62	0	0	-63	-64	0.000	0.000
125	27	-28	-38	-50	-56	0.000	0.000
126	19	-16	-18	-30	-37	0.027	0.022
128	333	0	0	-333	-325	0.000	0.006
142	229	-30	-10	-139	-139	0.063	0.063
148	260	-20	-23	-268	-286	0.003	0.002
154	175	-2	-2	-177	-174	0.000	0.001
161	121	15	18	-72	-46	0.087	0.132
173	16	-12	-22	-32	-42	0.000	0.000
180	17	-6	-6	-20	-22	0.000	0.000
185	18	-2	-2	-19	-15	0.000	0.028
187	21	-2	-2	-22	-25	0.000	0.000
189	890	0	0	-886	-874	0.001	0.004
192	7	0	0	-8	-11	0.000	0.000
194	4	-6	-12	-13	-18	0.000	0.000
202	627	-15	100	-427	-427	0.057	0.057
228	65	0	0	-66	-71	0.000	0.000
240	131	0	0	-131	-128	0.000	0.004
282	28	-6	-10	-38	-43	0.000	0.000
10001	135	-2	-6	-141	-142	0.000	0.000
10002	321	-3	-5	-326	-325	0.000	0.001
10003	73	-10	-12	-83	-88	0.000	0.000
10004	18	-38	-44	-50	-66	0.000	0.000

TREC-6 filtering results for TEKLIS

4 Routing task

For applying TEKLIS to the routing task of TREC we used the normalized fuzzy set probability described in chapter 2 as the ranking function. Otherwise we didn’t changed the algorithm.

Our results on the routing task (category B) were as shown below:

Total number of documents over all queries
Retrieved: 46000
Relevant: 3499
Rel_ret: 2580

Interpolated Recall - Precision Averages:

at 0.00	0.4988
at 0.10	0.3484
at 0.20	0.2765
at 0.30	0.2388
at 0.40	0.1990
at 0.50	0.1682
at 0.60	0.1497
at 0.70	0.1243
at 0.80	0.0837
at 0.90	0.0472
at 1.00	0.0101

Average precision (non-interpolated) over all rel docs:
0.1774

Precision:

At 5 docs:	0.3000
At 10 docs:	0.3022
At 15 docs:	0.2884
At 20 docs:	0.2728
At 30 docs:	0.2652
At 100 docs:	0.2115
At 200 docs:	0.1577
At 500 docs:	0.0920
At 1000 docs:	0.0561

R-Precision (precision after R (= num_rel for a query) docs retrieved):
Exact: 0.2053

5 Discussion

As mentioned earlier we tested our system on older TREC data with known relevance judgements. An analysis of the classification errors yielded some interesting results giving important hints for future improvements of TEKLIS:

- Computing a score normalized over the text length is not suitable for TREC data. We found many samples of very long documents, where the relevant passage was only one paragraph. In general the normalized score used by TEKLIS is only good if the documents are short or if a document should be categorized as a whole.

- For specific categories like the TREC topics it is easier to learn „very“ relevant words (comparable to concept words) than for more general categories, like economy, sports, politics, etc. But the ambiguity of these words can be greater for categorization purposes.

- The restriction of using single words without context is not sufficient for the TREC topics. We think it’s neccessary to enhance TEKLIS with the possibility of learning contexts for the most relevant words. First experiments showed an improvement of 10% in recall and precision on the filtering task.

Finally we tried to incorporate the topic descriptions into our training. Since TEKLIS is designed to learn from sample documents we didn’t used the topic descriptions for our official runs. In an experiment we used the topic descriptions as „artificial“ documents multiplying them 30 times for each category. After adding these „artificial“ samples to the training data we got significant improvements in our test results.

Reference

[1] T. Brückner, P. Suda, H.-U. Block and G. Maderlechner, „In-house Mail Distribution by Automatic Address and Content Interpretation“, 5th Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, Nevada, (1996), pp 72-74.