# MeSH based feedback, concept recognition and stacked classification for curation tasks

*Wessel Kraaij, †Marc Weeber, *Stephan Raaijmakers and †Rob Jelier

*TNO-TPD
P.O. Box 155, 2600 AD Delft
The Netherlands
{kraaij,raaijmakers}@tpd.tno.nl

†Department of Medical Informatics
Erasmus Medical Center
PO Box 1738,3000 DR Rotterdam
The Netherlands
{m.weeber,r.jelier}@erasmusmc.nl

**Abstract**

This paper reports about experiments carried out in the context of the genomics track at TREC 2004. Experiments were concentrated on two subtasks: the ad hoc retrieval task and the triage task. Experiments for the ad hoc task aimed at improving a standard full-text ad-hoc run (using a language modeling approach) by exploiting the manual classification of MEDLINE abstracts (the MeSH terms) for relevance feedback. The triage task was modeled as a standard classification task, using stacked classifiers and complex features, recognized by the Collexis IR engine.

## 1 Introduction

The research goals for the participation of TNO and Erasmus MC in this year's genomics track were restricted to submitting baseline runs for the ad hoc and triage task, with some minimal experimentation. The main ingredients for these runs were developed during previous projects. Still, this rather minimal effort includes some interesting experimentation, since the rich data sets offer ample opportunities for new research. Erasmus MC participated in the first issue of the genomics track in 2003 (Jelier et al., 2004). Erasmus has an active knowledge of the domain and has firm experience with using Collexis, a commercial tool for concept recognition, which was used for feature extraction in the triage task. TNO has a strong track record in language modeling based IR and machine learning techniques (Hiemstra and Kraaij, 1999; Kraaij et al., 2002; Kraaij, 2004; Hiemstra and Kraaij, 2005)
.

## 2 Ad Hoc task

The genomics Ad Hoc task is modeled as a standard TREC Ad Hoc evaluation of 50 topics created by experts. The particular interest of the task lies in the fact that the document database consists of MEDLINE abstracts that are annotated with a wealth of metadata, including MeSH headings. It was our research goal to investigate whether the structured

metadata could be exploited to improve upon a baseline run, using only the title and abstract fields.

## 2.1 Language modeling

The retrieval engine used for the Ad Hoc task is based on generative language models and uses cross-entropy between query and document models as main scoring criterion. It is the same engine that was used for previous TREC participations (e.g. WEB, Ad Hoc, SDR etc). Both documents and queries are represented as simple unigram language models. The parameters of the document language models are estimated by interpolating relative frequency of occurrence of the term $w$ in the document $D$ with the relative frequency of occurrence in the document collection $C$.

$$H(Q;D) = \sum_w P(w|Q) \sum_w \log(\lambda P(w|C) + (1-\lambda)P(w|D)) \tag{1}$$

Kraaij (2004) provides a more thorough description of this framework.

## 2.2 Combining full text and MeSH terms

Our approach to leverage information from MeSH headings was based on combining the results of two search queries on two distinct indices. The first index was built on the title and abstract fields of the MEDLINE records. The second index was built exclusively on the MeSH fields of the records.

Our combined search strategy consisted of four steps:

1. Search the abstract/title index, yielding our baseline run.
2. For each topic, extract the MeSH terms from the top $N$ documents of the baseline run. Concatenate these terms to form a MeSH query.
3. Search the MeSH index using the MeSH queries created in the previous step.
4. Interpolate the baseline run with the MeSH run. This is the combination run, which was submitted.

## 2.3 Experiments

The genomics ad hoc test collection consists of 10 years of MEDLINE records ranging from 1994 to 2003, 4591008 records in total (Hersh, 2004). For development, five sample topics with partial relevance judgements were provided. The test collection itself consists of 50 topics, formatted in TREC style, with a title, need and context field. We only worked with the full topics and applied both a standard stoplist plus a stoplist for query specific terminology which had been shown to be effective for previous ad hoc tasks.

**Baseline run** We carried out several experiments with the development topics. We found that stemming did hurt performance and simple pseudo relevance feedback techniques that had proven to be successful for the ad-hoc task were not very effective. Also, a document length prior was not effective, which was not surprising since the documents (abstracts) all have a quite similar length.

**MeSH run** Our main goal was to try to take advantage of the MeSH index terms, which were added by experts. The problem we faced, was that the supplied topics did not specify any relevant MeSH terms, so we had to use the document collection to infer MeSH terms for each topic.

We applied a non-standard tokenizer for the MeSH index: in order to preserve phrases in complex MeSH terms, blanks were replaced by underscores. We experimented with preserving the special character '*' and with keeping compound MeSH terms intact or splitting them. On the basis of the development data, we decided to ignore the asterisk and to split compound MeSH terms on the ',' character, e.g.,

```
'Endoplasmic Reticulum, Rough' => 'Endoplasmic_Reticulum' 'Rough'
```

MeSH queries were inferred from the baseline run, by a simple concatenation of the MeSH terms of the top 3 documents returned. This parameter was found optimal on the development data.

**Combination run** The combination run consisted of a simple linear interpolation of the baseline run with the MeSH run. An interpolation parameter of 0.8 (favouring the baseline run) was optimal for the development data.

### 2.4 Results

The most striking result is that our systems perform much better on test data than development data. This is probably due to the much more extensive pool which was judged for the test topics. The combination run performs better for both topic collection, although differences are small for the test topics.

| condition | development data | test data |
|---|---|---|
| baseline(1) | 0.1069 | **0.3196** |
| MeSH queries(3) | 0.0614 | 0.1313 |
| combination(4) | 0.1163 | **0.3247** |

Table 1: Results on development data (5 topics) and test data (50 topics) (mean average precision). Results of the official runs are printed in bold

Figure 1 shows a comparison of our official runs with the median performance. The baseline performs well above median. The combination run is slightly better, but the added value of the MeSH run is not entirely convincing. A first shallow analysis seems to suggest that the MeSH run mainly helps to improve precision as it hardly contributes unique relevant documents to the combination run.

Unfortunately there are no manual MeSH based queries available, which would enable a direct comparison between full text abstract search and controlled term search. The automatically generated MeSH queries have not been judged by experts and could therefore be far from optimal.

## 3 Triage task

The triage task is concerned with deciding whether a document merits manual classification in a gene ontology or not.

We have approached the TREC Genomics Triage task as a classification task. Based on features extracted from the Medline citations, a memory-based learning algorithm (MBL) has been trained to decide whether a certain document should be considered for manual annotation or not.

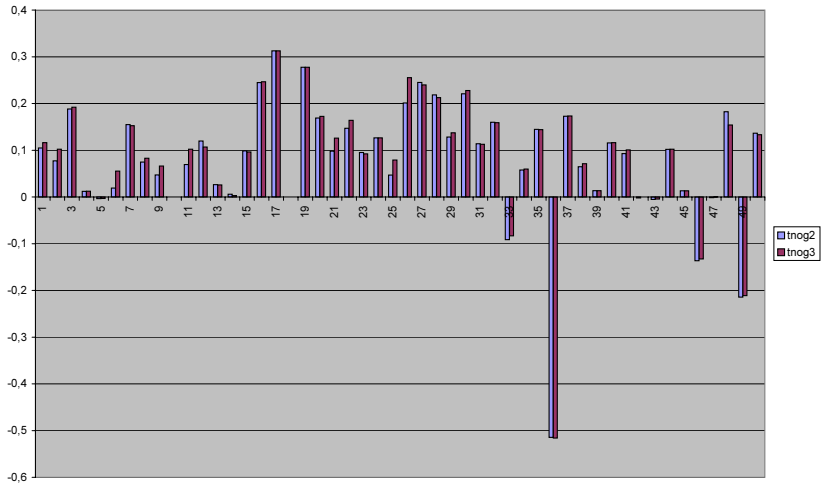


Figure 1: Comparison with median results

## 3.1 Stacked classification

Stacked classification (stacking; (Wolpert, 1992)) is a form of ensemble learning where a *meta learner* combines the predictions of *base learners* for a certain classification task into a single prediction. Basically, a number of uncorrelated base learners, each either performing a different machine learning algorithm or employing a different feature space, is applied to a certain classification problem. When the number of base learners is small, say under 10, majority voting over the predictions of these learners is not feasible. Instead, a meta learner can be constructed that combines the features and classifications of each of the base learners into one complex representation. This implies that the classifications of the base learners are now treated as features by the meta learner. Given the following descriptions of $n$ base learners (commonly referred to as $L0$-classifiers) for a classification problem $F \rightarrow C$ ($F$ a feature space, and $C$ a set of classifications)

$$\begin{aligned} D(L0_1) &= f_1, \ldots, f_n, c_k \\ &\vdots \\ D(L0_n) &= f_1, \ldots, f_n, c_l \end{aligned}$$

a meta learner $L1$ would use the following information to arrive at its prediction for a particular instance: $D(L0_1) \oplus \ldots \oplus D(L0_n)$. This is just the juxtaposition of features and classes of the various $L0$ classifiers. In this sense, an $L1$ classifier is an *arbiter*, weighting the features and classifications of the subordinate classifiers.

## 3.2 Experiments

For the ad hoc task, there were 4 different feature spaces: [DESCRIPTION: Marc]. Available training data was split into training, test and development sets. Our machine learning model is Memory-Based Learning (MBL; @ref@: Daelemans et al), a variant of k-nearest

distance learning with information gain-based feature weighting. Four $L0$ classifiers were created, each one using a different feature space. The four $L0$ classifiers were applied to the development test data, and their features and predictions for these data were used to train a memory-based L1 classifier. The L0 classifiers all used a $k$ of 1 due to the numerical nature of the features, and infogain feature weighting. The trained L1 classifier, also with $k = 1$ and infogain feature weighting, was subsequently applied to the test data.

## 3.3 Features

While the full texts for both the training and the test collections were available, we have only looked at medline citation information, notably the title, abstract, and MeSH headings. We concatenated these fields to one text. This choice was based on the assumption that human annotators only use this (limited) information to decide whether a paper should be annotated/curated or not We applied the Schwartz and Hearst (2003) abbreviation expansion algorithm to resolve abbreviations, and thus potentially reducing gene symbol ambiguity. Using Collexis indexing technology ((van Mulligen et al., 2000), see also http://www.collexis.com), we identified biomedical concepts from 5 thesauri: MeSH, Mouse genes (MGI), and three GO thesauri, viz the GO function, component, and process thesaurus. Based on the extracted concepts, we have compiled different feature subsets:

**1** MeSH: The MeSH thesaurus consists over approx 20,000 concepts. Using the concepts as features directly is not feasible. Instead, we employ the UMLS semantic network that classifies all concepts into one or more semantic types. There is a total of 134 semantic types (ST). For each citation, we counted the number of concepts belonging to the 134 STs. (134 features)

**2** MeSH: Additional to the semantic type classification of concepts, there is a higher level categorisation of the 134 semantic types into 15 different semantic groeps (SG). For instance, both STs of "Lipid" and "Pharmacologic Substance" fall into the SG of "Chemicals & Drugs". See (McCray et al., 2001) for more information. We counted the number of concepts belonging to the 15 SGs for each citation (15 features)

**3** GO: We used the three Gene Ontology thesauri of GO function, GO component, and GO process. We counted the number of GO concepts per citation (3 features)

**4** MGI: We compiled a thesaurus of mouse gene names from the MGI (Mouse Genome Informatics) database. We counted the number of indexed mouse genes (1 feature)

**5-8** We normalized the counts in the sets of 1) to 4) by dividing the count by the number of concepts found in a citation for each specific thesaurus

**9** Journal name (1 feature)

We have aggregated different subsets into four final feature sets:
a) ST_counted: 1, 3,4,9 b) ST_weighted: 5, 7, 8, 9 c) SG_counted: 2-4, 9 d) SG_weighted: 5-7, 9

## 3.4 Results

Results displayed a satisfactory precision, but disappointing recall:

```
Run: EMCTNOT1
Precision: 0.2000
Recall: 0.0143
F-score: 0.0267
```

```
Normalized Utility: 0.0114


Statistics computed over 59 triage runs.

                      Best  Median Worst
                     ------ ------ ------
Precision            0.2309 0.1360 0.0713
Recall               0.9881 0.5571 0.0143
F-score              0.2841 0.1830 0.0267
Normalized Utility   0.6512 0.3425 0.0114
```

A very preliminary conclusion is that the abstract alone does not provide enough information. Also, it may be better to use the subsets mentioned above for single classifiers, and use a meta-classifier to aggregate on more different level-1 classifiers instead of concatenating different subsets into one set of features. A meta-classifiewr on the four final featureset seems not useful as there is a lot over overlap in feature subsets. A further thorough error analysis is planned to explain the disappointing recall in the triage task.

# 4 Conclusion

A simple and proven approach to ad hoc search based on generative language models set a quite good baseline for the ad hoc search task on 4.5 million MEDLINE records. Our attempts to improve the baseline full text run with a feedback run using MeSH index descriptors were successful, but the improvement is rather small. A preliminary analysis showed that our technique mainly improved precision and did not improve recall in a substantial way.

Our triage system was less successful, while displaying satisfactory precision, its recall proved quite weak. Further analysis is planned to explain these results.

# References


Hersh, B. (2004). Trec 2004 genomics final track protocol. `http://medir.ohsu.edu/~genomics/2004protocol.html`.

Hiemstra, D. and Kraaij, W. (1999). Twenty-one at TREC-7: Ad hoc and cross language track. In Voorhees, E. M. and Harman, D. K., editors, *The Seventh Text REtrieval Conference (TREC-7)*, volume 7. National Institute of Standards and Technology, NIST. NIST Special Publication 500-242.

Hiemstra, D. and Kraaij, W. (2005). *the TREC book*, chapter Language models at TREC. MIT press. forthcoming.

Jelier, R., Schuemie, M., van der Eijk, C., Weeber, M., van Mulligen, E., Schijvenaars, B., Mons, B., and Kors, J. (2004). Searching for generifs: Concept-based query expansion and bayes classification. In *Proceedings of TREC 2003*. NIST special publication SP 500-255.

Kraaij, W. (2004). *Variations on language modeling for information retrieval*. PhD thesis, University of Twente. `http://dis.tpd.tno.nl/mmt/pubs/wkthesis.pdf`).

Kraaij, W., Westerveld, T., and Hiemstra, D. (2002). The importance of prior probabilities for entry page search. In Beaulieu, M., Baeza-Yates, R., Myaeng, S. H., and Järvelin, K., editors, *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002)*. ACM Press.

McCray, A., Burgun, A., and Bodenenreider, O. (2001). Aggregating umls semantic types for reducing conceptual complexity. In *Medinfo 2001*, pages 216–220.

Schwartz, A. and Hearst, M. (2003). A simple algorithm for identifying abbreviation definitions in biomedical text. In *Pac Symp Biocomput. 2003*, pages 451–462.

van Mulligen, E., Diwersey, M., Schmidt, M., Buurman, H., and Mons, B. (2000). Facilitating networks of information. In *Proc AMIA Symp. 2000*, pages 868–872.

Wolpert, D. (1992). Stacked generalization. *Neural Networks*, (5):241–259.