The Ferrety Algorithm for the KDD Cup 2005 Problem

Zsolt T. Kardkovács, Domonkos Tikk, Zoltán Bánsághi
Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics
1117 Budapest, Magyar Tudósok krt. 2.
Hungary
kardkovacs@tmit.bme.hu, tikk@tmit.bme.hu, empzoooli@gmail.com

ABSTRACT

In this paper, we present a general solution for the KDD Cup 2005 problem. It uses the Internet as source of knowledge and extends it to categorize very short (less than 5 words) documents with reasonable accuracy. Our approach consists of three main parts: i.) a central knowledge filter ii.) an on-demand web crawler and iii.) a very efficient categorizer system. Our solution obtained Creativity and Precision Runner-up Awards at the competition. The main idea of Ferrety Algorithm can be generalized for mapping one taxonomy to another if training documents are available.

1. PRELIMINARIES

The KDD Cup 2005[1] set focus on query categorization. The participants had to classify 800,000 internet user search queries into 67 categories specified by the organizers.
For brevity, in this paper we will use the following denotation:

- *query taxonomy*: the set of 67 query categories supplied by the organizers where queries should be classified to. Here we use the term "taxonomy", because we organized the 67 query categories into a 2-level hierarchy by grouping the original ones and adding a top level category above them.. Thus e.g. Computer, Entertainment, Living, etc. became top level categories and the original Computer/Hardware, Computer/Software, etc. became leaf-level categories.
- *query category*: a category of query taxonomy;
- *query*: an element of the 800,000 query set, if not stated otherwise;
- *stemmed query*: as above, but after each word of the query is stemmed;
- *Internet search engine (ISE) taxonomy*: a set of hierarchical organized categories that is provided for users to help finding documents and navigate on topics.

2. INTRODUCTION

Mapping documents into a fixed set of categories is a document categorization task. KDD Cup 2005 can obviously be considered as such a problem. Usually, document categorization requires supervised learning methods, however, in the case of KDD Cup 2005 there are some major differences compared to the well-known document categorization problems that exclude the simply use of an off-the-shelf text categorizer:

1. The documents are very short (up to five words for more than 90% of the corpus).
2. The corpus is very noisy (more than 30% of the corpus contains incorrectly represented non-English documents or trash queries in particular).
3. The number of sample queries (only 111 pieces) is two small to serve as training documents.

As a consequence, the set of sample queries can only be used for evaluation, hence no training corpus was given. Nevertheless, we argue for the strong necessity of supervised learning because

- The queries contain large number of Named Entities (NEs) that can be processed efficiently (if it is possible in any way) by means of dictionaries, lexicons or training corpora. The presence of NEs, respected to the whole corpus, is high enough to use such a resource.
- If a useable set of categories existed for this problem, it still could not be mapped directly to the KDD Cup 2005 problem, since there is not any given semantics for query categories. The lack of semantics questions the validity of any mapping (assuming that there is a valid correspondence between the taxonomies).

Therefore, here we propose a new technique (named Ferrety algorithm) to overcome these problems, which consists of the following steps (see also Figure 1):

1. Find source: Determine the proper semantics for query categories by creating a basic dictionary and an ontology for them. Create valid mappings between query taxonomy and existing Internet search engine taxonomies by means of category semantics just determined.
2. Stem queries: Preprocess all queries by performing stemming on them. The result is the set of stemmed queries.
3. Query the Web: Exploiting the connections between query taxonomy and ISE taxonomies gather relevant training documents to (stemmed) queries by posting them to ISEs.

[1] `http://kdd05.lac.uic.edu/kddcup.html`

4. Parse results: Parse the results of Step 3.

5. Ask HITEC: Feed the parsed results to a text categorizer as training set for query taxonomy. We applied HITEC for text categorization purpose (see details in section 3.4 and in works [1; 3; 4]).

6. Run the trained text categorizer on all queries. Rank the obtained results by Internet querying and text categorizing and determine the top 5 categories for each query.

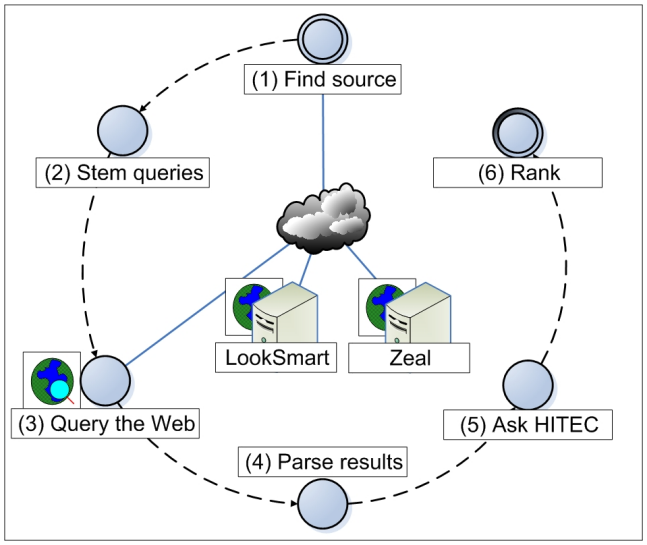

Figure 1: The steps of the Ferrety Algorithm

In the next sections, we discuss these steps in details.

3. ON THE FERRETY ALGORITHM

In this section, we describe briefly the overall concept of our solution. Later on, we discuss the steps our solution more detailed.

3.1 Internet as a Knowledge Base

The Ferrety Algorithm uses the Internet as a knowledge base by means of asking other search engines what they may know about the given query. That is, the Ferrety Algorithm can be treated as a meta-search engine. Actually, in this experiment it used only two engines: LookSmart (`search.looksmart.com`) and Zeal (`www.zeal.com`). Results of these engines were processed locally. LookSmart and Zeal (L&Z) engines are using very similar taxonomies so combining them was an obvious and more or less easy task.

Notwithstanding, rebuilding LookSmart or Zeal would not be an efficient solution and would have no sense, as these taxonomies are very large and subtle. They are efficient in finding pages and region related resources if the user has a determined purpose to query the web. Obviously, that is why taxonomies cover the most common user queries. On the other hand, a full-fledged model of the world can not be represented by trees as proposed by these engine makers. Therefore, the Ferrety Algorithm only *uses* the public interface to determine *some* initial answers for the user queries, and after that it tries to improve them.

Moreover, L&Z have a very clear and useful description on each category that determines the proper semantics for the given category. We created our basic dictionary and ontology based on this phenomenon.

3.2 On the Semantics of Categories

Since we strongly believed that the KDD Cup 2005 problem required a supervised learning method, we acquired our training data as follows. First, we stemmed all queries of the KDD Cup 2005 corpus by Porter stemmer. Then we posted the stemmed queries to Internet search engines for relevant answers. In general, we prefer to use a special set of clue words to find the most relevant taxonomy in order to get a corpus with a reasonable quality. Obviously, in the case of KDD Cup 2005, the corpus itself was the most adequate one to find the necessary taxonomy.

L&Z results have two main parts: *local categories* to which the query may belong (if any) and *relevant contexts* of the words contained in the query. Each category has a *semantic description* that describes the content of the category with a few words. We looked at the semantic description of local categories and we saved their default path from the root. As a result, we obtained semantic descriptions (real definitions) to all L&Z-relevant categories and a default hierarchy of categories. From now on, we call this corpus Basic Corpus ($\mathcal{BC}$), and its categories $\mathcal{BC}$-categories.

3.3 Building and Mapping Taxonomies

The taxonomy building procedure was designed to determine the maximum relevant set of words which may correspond to a given query category. It assumes that the query category name most accurately describes the information belonging to that particular query category. If the name is the most proper description, then all synonyms are also supposed to be valid terms to that category. That is why our procedure extends query category names by their WordNet synonyms. We call this step a semantic closure. Let

$$W(0) = \bigcup_i w_i(0)$$

stand for this initial set of partitions where $w_i(0)$ denotes the semantic closure of the ith query category.

We applied $W(0)$ to find relevant $\mathcal{BC}$-categories, i.e. we searched $\mathcal{BC}$-categories with semantic description containing the given set of words. Let $\mathcal{C}_i^0 \subseteq \mathcal{BC}$ be the subset of the Basic Corpus determined by $w_i(0)$. The Ferrety Algorithm calculates the well-known tf-idf measure (see e.g. [2]) for the words of the descriptions in $\mathcal{C}_i^0$, which have frequency greater then ω in at least one $\mathcal{C}_i^0$, and occur no more than in α $\mathcal{C}_i^0$. Let A^0 denote the set of words that holds this property. Then we can apply the recursive formula:

$$w_i(n+1) = w_i(n) \cup \{a | a \in A^n \cap \mathcal{C}_i^n\} \quad (n = 0, 1, \ldots) \quad (1)$$

We also use this step for the higher level categories, e.g. Computers, Living, Entertainment, etc. For this level, we create the union of sub-levels, and apply the above mentioned procedure. This step is important, because the limitation in idf factor might eliminate words that determine e.g. the valid top level context of Computers. Later on, if a query is determined to be in Computers, however, none of its sub-levels proved to be appropriate, then it will be categorized as "other" (here: Computer/Other).

One can easily see that by the finiteness of words occurring in finite many semantic descriptions the algorithm described by Equation (1) always terminates. Categories of

$$\mathcal{C}^- = \mathcal{BC} \setminus \bigcup_i \mathcal{C}_i^N$$

are classified using the L&Z default hierarchy in a way, that we propagate mapping results downward in the hierarchy until an element in $\mathcal{C}^+ = \mathcal{BC} \setminus \mathcal{C}^-$ (a *flagged category*[2]) is found, or the tree ends. Here $A \setminus B$ means elements of A that are not included in B (set difference). Note that, at this point we obtained both the basic dictionary (it is the union of w_i^Ns), and the mapping between L&Z and query taxonomies since $\mathcal{C}_i^N$s are query categories while their elements belong to L&Z categories. The flowchart and the pseudo-code of the algorithm is illustrated in Figure 2 and Figure 3, resp.

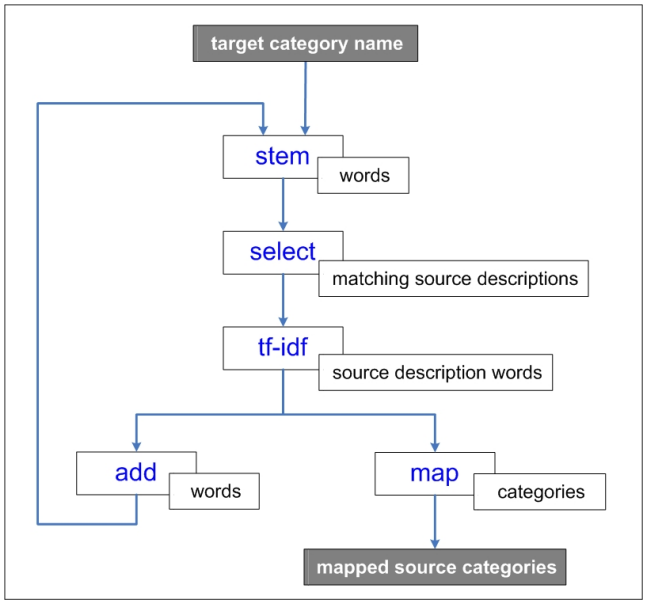

Figure 2: The steps of taxonomy mapping

Note that the algorithm described in Figure 3 can be applied to connect a *source* taxonomy and a *target* taxonomy, if the source taxonomy has semantic description for categories. This condition can easily be fulfilled if there are available training documents filed in the source taxonomy. The semantic description of a category can be obtained from the most frequent terms of assigned training documents. A profile or a prototype vector of a category can also be considered as semantic description.

3.4 Training and Categorization

We applied HITEC [1; 3; 4] categorizer with the Internet to determine categories for the queries. The current version of HITEC implements a neural network based learning algorithm that creates a prototype document for each category based on the training samples. When categorizing unknown documents, HITEC compares them to prototype documents of categories downward in the taxonomy and determines the most relevant ones. There are numerous parameters of both training and categorization to control the depth and the width of the searching path of the categorization. For more details see the references.

[2]A category is flagged in L&Z-taxonomy if it has a corresponding query category

3.4.1 On the Training Sets

Because the queries themselves contains very few words we found that the feature set (size of dictionary) might not be large and descriptive enough to determine the correct categories for unknown queries, therefore, we supplemented the training set and created 4 different alternatives for training. When explaining training sets, we refer the Reader to Subsection 3.2 for notations of L&Z result pages.

The first 3 alternatives assign training data to L&Z categories by means of asking the search engines with stemmed queries. The training data is the query itself or its augmented version as described below. The training data is assigned to such L&Z categories that occurs at least once among the local categories on the ISE's result page. The link between L&Z and query categories is established by the taxonomy mapping described in Section 3.3. The last alternative of training set assigns training data L&Z categories directly, that is without using query information.

Q Query: This is the simplest case, the training data is the stemmed query itself. This information is assigned to local categories of L&Z taxonomy that occurred at least ones in the search engine's result page.

WQ Weighted Query: Same as the previous, but the text of the stemmed queries is assigned as many times to the local categories of L&Z taxonomy, as the given category occurred on the search engine's result page. That is, if on the result page of q query c local category appears twice, the text of q is assigned twice to c.

T Text: The text of the stemmed queries is augmented by the context of the query in the following way. We downloaded L&Z result pages and extracted ASCII text from the HTML pages. Stemmed queries were "enriched" with this text, thus we gained additional relevant information to the queries. The assignment of this training data to categories was done analogously as in the above two cases.

C Category info: Short semantic description of L&Z categories provided by the ISEs were added to enrich the information about categories. Note that, these information are not assigned directly to queries but to L&Z categories. The short description contains a title and a descriptive sentence about the category. We assigned different weights to these text fields (see also Table 1). We applied them only for flagged categories.

3.4.2 Feature selection

One of the most crucial parts of the training is the determination of the feature set. One has to find the optimal balance between discarding rare terms and keeping discriminative ones in the dictionary. In HITEC we have two simple parameters to control the dictionary size ($|\mathcal{D}|$): a lower limit for the minimal occurrence (d_1) of a term in the entire (training) corpus, and a maximum allowed value (d_2) for the overall distribution of a term over the corpus. These parameters are related to the TFIDF frequency scheme: d_1 and

input: root node of a subtree and the tree structure
output: a map array between source and target categories

```
function mapCategories( Node Root, Graph Tree ) begin
    if ( Root is leaf )
        return nil;
    foreach S in children of the Root in Tree
    begin
        let NewWords[S] := { stemmed words of S and their synonyms };
        let CandCat[S] := Empty;
        let Words[S] := Empty;
    end
    while ( NewWords != Words )
    begin
        let Words := NewWords;
        foreach S in children of the Root in Tree
        begin
            let NewCandCat[S] := CandCat[S] + { source categories with any word in Words[S] };
            let NewWords[S] := { stemmed words in NewCandCat[S]'s descriptions with frequency > tf };
        end
        let NewWords := array of set of words in which words do not occur in more than x keys,
                                                        where x is the threshold for idf;
        let CandCat := array of set of categories in which categories occur under a single key;
    end
    return CandCat;
end
```

Figure 3: The pseudo-code of the taxonomy mapping algorithm

d_2 can be considered as lower threshold for TF and upper threshold for DF, respectively.

For the given problem, especially when using training sets Q, T or C (or their meaningful combinations), we applied very low d_1 (2–5), and very high d_2 (over 0.5) because the entire corpus contains relatively few terms, and terms occur very rarely. We also found that d_1 parameter should be kept low for training set C, because a lot of discriminative words occurred only 1 or 2 times in the entire corpus. We applied two weight factors when training set C was concerned: w_t is for the title and w_d for the description of the category, resp. In case of the largest training set (Q + T + C), we got 1080k terms in the original dictionary (after stemming). We decreased its size to 809k. With training set Q+C, we got only 149k terms in the dictionary. We tabulated some significant feature selection runs in Table 1.

Table 1: The size of dictionary in terms of the parameters of feature selection

Run	d_1	d_2	w_t	w_d	$\lvert\mathcal{D}\rvert$	Training set
R1	2	0.7	5	3	888565	Q+T+C
R2	5	1.0	10	3	149792	Q+C
R3	5	1.0	10	3	809165	Q+T+C
R4	0	1.0	10	3	1088171	WQ+T+C
R5	1	1.0	10	3	763872	T+C

3.4.3 Parameters of training

At training we fixed the number of iteration for 5, which had been found quasi-optimal for other large corpora (e.g. Reuters Corpus Volume 1). We used all training documents for the training.

There are two important parameters to control the size of the result set of a categorization. By max-variance ($v_{\max}$) one can specify the minimal deviation (ratio) of a node to be considered for further search w.r.t. the maximal confidence value at a given hierarchy level. Setting this value low (around 0.5), one can have many firing categories at each level of the taxonomy. By threshold (θ), one can set the minimal confidence value of a node. Setting this value low ($0.05 \sim 0.15$) results in better recall values and more result categories. With the help of these two parameters, we were able to make a trade-off between recall and precision.

For the current task, our goal was to obtain large result set for the categorization, therefore we set $v_{\max} = 0.5$ and $\theta = 0.1$.

HITEC provides possibility to use different training corpora for dictionary creation and for training. We exploited this feature and in most run disregarded the Q training set, because the using of Q at training decreased the quality of categorization. This fact can be explained by arguing that the query text are two short for effective categorization. On the other hand, Q is useful at dictionary creation since it could raise the importance of certain occurring terms in the entire Q+T+C training corpus.

The learning capability of some settings—i.e. how HITEC could learn the training data—is indicated in Table 2. Figure 4 shows the effect of learning during iterations in terms of HITEC's inner quality measure. The efficiency measures shown in the table is obtained by testing the algorithm on indicated training sets (see also Section *3.4.1*).

3.4.4 Evaluation

We evaluated various feature selection and training settings by means of the 111 sample queries (see also Figure 5).

Table 2: Validation results of certain runs (same notation used as in Table 1)

Run	$v_{\max}$	θ	F_1	Precision	Recall	Train
R1	0.5	0.1	0.655	0.586	0.741	T+C
R2	0.5	0.1	0.532	0.459	0.634	Q+C
R3	0.5	0.1	0.517	0.446	0.616	T+C
R5	0.7	0.2	0.326	0.28	0.391	Q+T+C

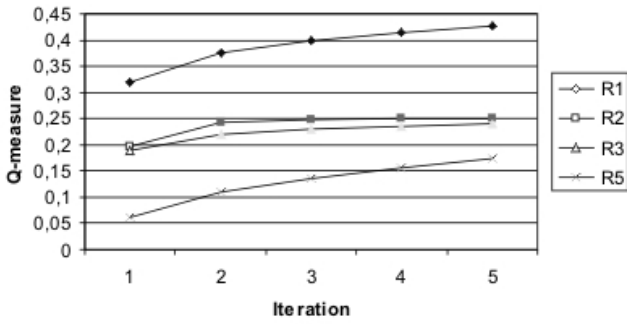

Figure 4: The effect of learning during iterations in terms of HITEC's inner quality measure

In the evaluation we also took into consideration the results obtained by passing the sample queries to Looksmart and processing them analogously as training set (*training result*). Comparing the results for the 111 samples, we found that results categorized by HITEC were considerably worse than training results regardless of the parameters of feature set and training. This is not at all surprising because in HITEC's result two errors are cumulated: the error of the training set (that can also be considerable) and the error of HITEC's training and categorization. There are also quite a few query words among the 111 queries that do not occur any more in the entire corpus (e.g. *aldactone*).

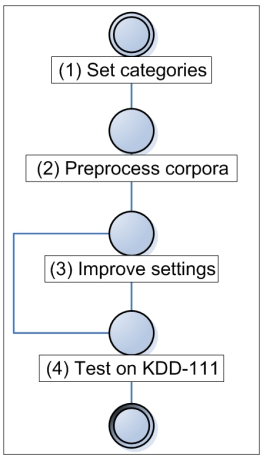

Figure 5: Evaluation by means of HITEC

The efficiency of training was checked in terms of two factors. First, on the 111 samples queries, and second, on the queries of the training set used as validation (F_1 measure). We found that the training set (T + C) gave the most promising result with $d_1 = 2$, $\theta = 0.1$ and $v_{\max} = 0.5$. Unfortunately, R1 run finished after the submission deadline, so we used the result of R2 run for submission.

3.5 Evaluation of submitted Results

The submitted results were aggregated from two sources:

1. for the queries in the training set (cca. 400k queries) we submitted the results obtained after processing L&Z pages;
2. for those queries that did not produce useable results from Looksmart and Zeal search engines, we submitted results of HITEC's categorization.
3. There were about 80k queries—most probably mainly trash and foreign ones—that did not result any categories at all.

The above sets were selected based on validation performed on 111 sample queries and on the consideration that HITEC's would rather increase than decrease the categorization error on training set.

Because of the lack of time, we had no chance to do fine tuning on HITEC's setting. Hence, our approach achieved 0.340883 and 0.34009 precision and F_1-measure in the evaluation, resp, as determined by the organizers. We reckon that it could be raised at least a few percent if more time would have left for tuning, and could submit the HITEC results on R1 run. In the near future, we will investigate the effect of HITEC's setting on query categorization results, and the robustness of Ferrety algorithm.

4. CONCLUSION

In this paper we presented an algorithm to solve the KDD Cup 2005 problem. The taxonomy building procedure applies interesting ideas (set of clue words, modified tf-idf calculation) to determine the proper category of very short description. We combined a web search based categorizer and a taxonomy mapper for limited training sets. The taxonomy mapper that we developed for this particular competition can be applied for general use in connecting taxonomies. We feel that our approach were very successful considering that (i) we participated first time at the competition (ii) started the problem from scratch (apart from HITEC), and (iii) our team had the fewest members among the winning teams.

5. ACKNOWLEDGEMENTS

Our thanks to KDD Cup 2005 organizers, Ying Li (Microsoft) and Zijian Zheng (Amazon) for the interesting and successful competition. We would also like to thank György Biró (Textminer Ltd., Budapest) for the use of HITEC autocategorizer software. Domonkos Tikk was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Science.

6. REFERENCES

[1] HITEC categorizer online. `http://categorizer.tmit.bme.hu`.

[2] G. Salton and M. J. McGill. *An Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[3] D. Tikk, Gy. Biró, and J. D. Yang. A hierarchical text categorization approach and its application to FRT expansion. *Australian Journal of Intelligent Information Processing Systems*, 8(3):123–131, 2004.

[4] D. Tikk, Gy. Biró, and J. D. Yang. Experiments with a hierarchical text categorization method on WIPO patent collections. In N. O. Attok-Okine and B. M. Ayyub, editors, *Applied Research in Uncertainty Modelling and Analysis*, number 20 in International Series in Intelligent Technologies, pages 283–302. Springer, 2005.

About the authors:

Zsolt T. Kardkovács is an assistant lecturer at Department of Telecommunications and Media Informatics (DTMI) at Budapest University of Technology and Economics (BUTE). His research interests include (deductive and relational) databases, logic, knowledge representation, natural language processing, and human computer interaction. He is expected to get his Ph.D. degree early next year.

Domonkos Tikk is a Senior Researcher at DTMI, BUTE. He has received his Ph.D. in 2000 from the same institution in fuzzy systems. His research covers text mining, natural language processing (NLP), internet search engines, pattern recognition, fuzzy systems and soft computing techniques. Recently he is focusing on hierarchical text classification and NLP related problems. He is acted as the major coach of this KDD Cup team.

Zoltán Bánsághi is a graduate student at BUTE. His field of interest includes text mining, natural language processing and programming.