Query Classification using Wikipedia's Category Graph

Milad AlemZadeh
Centre for Pattern Analysis and Machine Intelligence, University of Waterloo, Waterloo, Ontario, Canada
Email: malemzad@uwaterloo.ca

Richard Khoury
Department of Software Engineering, Lakehead University, Thunder Bay, Ontario, Canada,
Email: richard.khoury@lakeheadu.ca

Fakhri Karray
Centre for Pattern Analysis and Machine Intelligence, University of Waterloo, Waterloo, Ontario, Canada
Email: karray@uwaterloo.ca

***Abstract*— Wikipedia's category graph is a network of 300,000 interconnected category labels, and can be a powerful resource for many classification tasks. However, its size and the lack of order can make it difficult to navigate. In this paper, we present a new algorithm to efficiently exploit this graph and accurately rank classification labels given user-specified keywords. We highlight multiple possible variations of this algorithm, and study the impact of these variations on the classification results in order to determine the optimal way to exploit the category graph. We implement our algorithm as the core of a query classification system and demonstrate its reliability using the KDD CUP 2005 and TREC 2007 competitions as benchmarks.**

***Index Terms*—Keyword search, Natural language processing, Knowledge based systems, Web sites, Semantic Web**

I. Introduction

Query classification is the task of Natural Language Processing (NLP) whose goal is to identify the category label, in a predefined set, that best represents the domain of a question being asked. An accurate query classification system would be beneficial in many practical systems, including search engines and question-answering systems. Query classification shares some similarities with other categorization tasks in NLP, and with document classification in particular. However, the challenge of query classification is accentuated by the fact that a typical query is only between one and four words long [1], [2], rather than the hundreds or thousands of words one can get from an average text document. Such a limited number of keywords makes it difficult to select the correct category label, and moreover it makes the selection very sensitive to "noise words", or words unrelated to the query that the user entered for some reason such as because they didn't remember a correct name or technical term to query for. A second challenge of query classification comes from the fact that, while document libraries and databases can be specialized to a single domain, the users of query systems expect to be able to ask queries about any domain at all [1].

This paper continues our work on query classification using the Wikipedia category graph [3], [4]. It refines and expands on our previous work by studying multiple different design alternatives that similar classification systems could opt for, and considers the impact of each one. In contrast with our previous papers, the focus here is not on presenting a single classification system, but on implementing and comparing multiple systems that differ on critical points.

The rest of the paper is organized as follows. Section 2 presents overviews of the literature in the field of query classification with a special focus on the use of Wikipedia for that task. We present in detail our ranking and classification algorithm in Section 3, and take care to highlight the points where we considered different design options. Each of these options was implemented and tested, and in Section 4 we describe and analyze the experimental results we obtained with each variation of our system. Finally, we give some concluding remarks in Section 5

II. Background

Query classification is the task of NLP that focuses on inferring the domain information surrounding user-written queries, and on assigning to each query the best category label from a predefined set. Given the ubiquity of search engines and question-handling systems today, this challenge has been receiving a growing amount of attention. For example, it was the topic of the ACM's annual KDD CUP competition in 2005 [5], where 37 systems competed to classify a set of 800,000 real web queries into a set of 67 categories designed to cover most topics found on the internet. The winning system was designed to classify a query by comparing its word vector to that of each website in a set pre-classified in the

doi:10.4304/jetwi.4.3.207-220

Google directory. The query was assigned the category of the most similar website, and the directory's set of categories was mapped to the KDD CUP's set [2]. This system was later improved by introducing a bridging classifier and an intermediate-level category taxonomy [6].

Most query classifiers in the literature, like the system described above, are based on the idea of mapping the queries into an external knowledge source (an objective third-party knowledge base) or internal knowledge source (user-specific information) to classify them. This simple idea leads to a great variety of classification systems. Using an internal knowledge source, Cao *et al.* [7] developed a query classifier that disambiguates the queries based on the context of the user's recent online history. And on the other hand, many very different knowledge sources have been used in practice, including ontologies [8], websites [9], web query logs [10], and Wikipedia [4], [11], [12].

Exploiting Wikipedia as a knowledge source has become commonplace in scientific research. Several hundreds of journal and conference papers have been published using this tool since its creation in 2001. However, while both query classification and NLP using Wikipedia are common challenges, to the best of our knowledge there have been only three query classification systems based on Wikipedia.

The first of these three systems was proposed by Hu et al. [11]. Their system begins with a set of seed concepts to recognize, and it retrieves the Wikipedia articles and categories relevant to these concepts. It then builds a domain graph by following the links in these articles using a Markov random walk algorithm. Each step from one concept to the next on the graph is assigned a transition probability, and these probabilities are then used to compute the likelihood of each domain. Once the knowledge base has been build in this way, a new user query can be classified simply by using its keywords to retrieve a list of relevant Wikipedia domains, and sorting them by likelihood. Unfortunately, their system remained small-scale and limited to only three basic domains, namely "travel", "personal name" and "job". It is not a general-domain classifier such as the one we aim to create.

The second query classification system was designed by one of our co-authors in [12]. It follows Wikipedia's encyclopedia structure to classify queries step-by-step, using the query's words to select titles, then selecting articles based on these titles, then categories from the articles. At each step, the weights of the selected elements are computed based on the relevant elements in the previous step: a title's weight depends on the words that selected it, an article's weight on the titles', and a category's weight on the articles'. Unlike [11], this system was a general classifier that could handle queries from any domain, and its performance would have ranked near the top of the KDD CUP 2005 competition.

The last query classification system is our own previous work, described in [4]. It is also a general classifier, but its fundamental principles differ fundamentally from [12]. Instead of using titles and articles to pinpoint the categories in which to classify a query like was done in [12], the classifier of [4] used titles only to create a set of inexact initial categories for the query and then explored the category graph to discover the best goal categories from a set of predetermined valid classification goals. This classifier also differs from the one described in this work on a number of points, including the equations used to weight and rank categories and the mapping of the classification goals. But the most fundamental difference is the use in this paper of pre-computed base-goal category distances instead of an exploration algorithm. As we will show in this paper, all these modifications are justified both from a theoretical standpoint and practically by improvements in the experimental results.

While using Wikipedia for query classification has not been a common task, there have been several document classification projects done using that resource which are worth mentioning. Schönhofen [13] successfully developed a complete document classifier using Wikipedia, by mapping the document's vocabulary to titles, articles, and finally categories, and weighting the mapping at each step. In fact, we used some of the mapping techniques he developed in one of our previous works [12]. Alternatively, other authors use Wikipedia to enrich existing text classifiers by improving upon the simple bag-of-words approach. The authors of [14] use it to build a kernel to map the document's words to the Wikipedia article space and classify there, while the authors of [15] and [16] use it for text enrichment, to expand the vocabulary of the text by adding relevant synonyms taken from Wikipedia titles. Interestingly, improvements are reported in the classification results of [13], [15] and [16], while only [14] reports worse results than the bag-of-words method. The conclusion seems to be that working in the word space is the better option; a conclusion that [14] also shares. Likewise, that is the approach we used in the system we present in this paper.

III. ALGORITHM

Wikipedia's category graph is a massive set of almost 300,000 category labels, describing every domain of knowledge and ranging from the very precise, such as "fictional secret agent and spies", to the very general, such as "information". The categories are connected by hypernym relationships, with a child category having an "is-a" relationship to its parents. However, the graph is not strictly hierarchic: there exist shortcuts in the connections (i.e. starting from one child category and going up two different paths of different lengths to reach the same parent category) as well as loops (i.e. starting from one child category and going up a path to reach the same child category again).

The query classification algorithm we propose in this paper is designed to exploit this graph structure. As we will show in this section, it is a three-stage algorithm, with a lot of flexibility possible within each step. The first

```
Input: Wikipedia database dump
1.  CG ← the Category Graph
    extracted from Wikipedia
2.  Associate to each category in CG
    the list of all titles pointing
    to it
3.  GC ← the set of Goal Categories
    identified in CG
4.  Dist(GC,CG) ← the shortest-path
    distance between every GC and all
    categories in CG
Input:  User query, CG
5.  KL ← Keyword List of all
    keywords in the user query
6.  TL ← Title List of all titles in
    CG featuring at least one word in
    KL
7.  KTW ← Keyword-Title Weight, a
    list containing the weight of a
    keyword from KL featured in a
    title from TL
8.  BC ← Base Categories, all
    categories in CG pointed to by TL
9.  CD ← Category Density for all BC
    computed from the KTW
10. BC ← top BC ranked by CD
Input: GC, DIST(GC,BC), CD
11. GS ← Goal Score of each GC,
    computed based on their distance
    to each BC and on CD
12. Return: top 3 GC ranked by GS
```

Figure 1. Structure of the three steps of our classification algorithm: the pre-processing step (top), the base category evaluation (middle), and the exploration for the goal categories (bottom).

stage is a pre-processing stage, during which the category graph is built and critical application-specific information is determined. This stage needs to be done only once to create the system, by contrast with the next two stages that are executed for each submitted query. In the second stage, a user's query is mapped to a set of base categories, and these base categories are weighted and ranked. And finally, the algorithm explores the graph starting from the base categories and going towards the nearest goal categories in stage 3. The pseudocode of our new algorithm is shown in Figure 1.

A. Stage 1: Pre-Processing the Category Graph

We begin the first stage of our algorithm by extracting the list of categories in Wikipedia and the connections between categories from the database dump made freely available by the Wikimedia Foundation. For this project, we used the version available from September 2008.

Furthermore, our graph includes one extra piece of information in addition to the categories, namely the article titles. In Wikipedia, each article is an encyclopedic entry on a given topic which is classified in a set of categories, and which is pointed to by a number of titles: a single main title, some redirect titles (for common alternative names, including foreign translations and typos) and some disambiguation titles (for ambiguous names that may refer to it). For example, the article for the United States is under the main title "United States", as well as the redirect titles "USA", "United States of America" and "United Staets" (common typo redirection), and the disambiguation title "America". Our pre-processing deletes stopwords and punctuation from the titles, then maps them directly to the categories of the articles and discards the articles. After this processing, we find that our category graph features 5,453,808 titles and 282,271 categories.

The next step in the graph construction category is to define a set of goal categories that are acceptable classification labels. The exact number and nature of these goal categories will be application-specific. However, the set of Wikipedia category labels is large enough to cover numerous domains at many levels of precision, which means that it will be easy for system designers to identify a subset of relevant categories for their applications, or to map an existing category set to Wikipedia categories.

The final pre-processing step is to define, compute and store the distance between the goal categories and every category in the graph. This distance between two categories is the number of intermediate categories that must be visited on the shortest path between them. We allow any path between two categories, regardless of whether it goes up to parent categories or down to children categories or zigzags through the graph. This stands in contrast with our previous work [4], where we only allowed paths going from child to parent category. The reason for adopting this more permissive approach is to make our classifier more general: the parent-only approach may work well in the case of [4] where all the goal categories selected were higher in the hierarchy than the average base category, but it would fail miserably in the opposite scenario when the base categories are parents of the goal categories. When searching for the shortest paths, we can avoid the graph problems we mentioned previously, of multiple paths and loops between categories, by only saving the first encounter of a category and by terminating paths that are revisiting categories. Finally, we can note that, while exploring the graph to find the shortest distance from every goal category and all other categories may seem like a daunting task, for a set of about 100 goal queries such as we used in our experiments it can be done in only a few minutes on a regular computer.

B. Stage 2: Discovering the Base Categories

The second stage of our algorithm as shown in Figure 1 is to map the user's query to an initial set of weighted base categories. This is accomplished by stripping the query of stopwords to keep only relevant keywords, and then generating the exhaustive list of titles that feature at least one of these keywords. Next, the algorithm considers each title t and determines the weight W_t of the keywords it contains. This weight is computed based on two parameters: the number of keywords featured in the title (N_k), and the proportional importance of keywords in the title (P_k). The form of the weight equation is given in

equation (1).

$$W_t = N_k P_k \quad (1)$$

The number of keywords featured in the title is a simple and unambiguous measure. The proportional importance is open to interpretation however. In this research, we considered three different measures of importance. The first is simply the proportion of keywords in the title (N_k / N_t, where N_t is the total number of words in title t). The second is the proportion of characters in the title that belong to keywords (C_k / C_t, where C_k is the number of characters of the keywords featured in the title and C_t is the total number of characters in title t). This metric assumes that longer keywords are more important; in the context of queries, which are only a few words long [1], [2], it may be true that more emphasis was meant by the user on the longest, most evident word in the query. The final measure of proportional importance is based on the word's inverted frequencies. It is computed as the sum of inverted frequencies of the keywords in the title to the sum of frequencies of all title words ($\Sigma F_k / \Sigma F_t$), where the inverted frequency of a word w is computed as:

$$F_w = \ln(T / T_w) \quad (2)$$

In equation (2), T is the total number of titles in our category graph and T_w is the number of titles featuring word w. It is, in essence, the IDF part of the classic term frequency-inverse document frequency (TFIDF) equation: $(N_w / N)\ln(T / T_w)$, where N_w is the number of instances of word w in a specific title (or more generally, a document) and N is the total number of words in that title. The TF part (N_w / N) is ignored because it does not give a reliable result when dealing with short titles that only feature each word once or twice. We have used this metric successfully in the past in another classifier we designed [12].

We can see from equation (1) that every keyword appearing in a title will receive the same weight W_t. Moreover, when a title is composed exclusively of query keywords, their weight will be the number of keywords contained in the title. The maximum weight a keyword can have is thus equal to the number of keywords in the query; it occurs in the case where a title is composed of all query keywords and nothing else.

Next, our algorithm builds a set of base categories by listing exhaustively all categories pointed to by the list of titles. This set of base categories can be seen as an initial coarse classification for the query. These base categories are each assigned a density value. A category's density value is computed by determining the maximum weight each query keyword takes in the list of titles that point to that category, then summing the weights of all keywords, as shown in equation (3). In that equation, D_i is the density of category i, and $W_t^{k,i}$ refers to the weight W_t of a title t that contains keyword k and points to category i. Following our discussion on equation (1), we can see that the maximum density a category can have is the square of the number of query keywords. It happens in the case where each keyword has its maximum value in that category, meaning that one of the titles pointing to the category is composed of exactly the query words.

$$D_i = \sum_k \max_t (W_t^{k,i}) \quad (3)$$

At the end of this stage of the algorithm, we have a weighted list of base categories, featuring some categories pointed to by high-weight words and summing to a high density score, and a lot of categories pointed to by only lower-weight words and having a lower score. In our experiments, we found that the set contains over 3,000 base categories on average. We limit the size of this list by keeping only the set of highest-density categories, as categories with a density too low are deemed to be too unrelated to the original query to be of use. This can be done either on a density basis (i.e. keeping categories whose density is more than a certain proportion of the highest density obtained for this query, regardless of the number of categories this represents, as we did in [4]) or on a set-size basis (i.e. keeping a fixed number of categories regardless of their density, the approach we will prioritize in this paper). When using the set-size approach, a question arises on how to deal with ties when the number of tied categories exceeds the size of the set to return. In our system, we break ties by keeping a count of the number of titles that feature keywords and that point to each category, and giving priority to the categories pointed to by more titles.

C. Stage 3: Ranking the Goal Categories

Once the list of base categories is available, the third and final stage of the algorithm is to determine which ones of the goal categories identified in the first stage are the best classification labels for the query. As we outlined in the pseudocode of Figure 1, our system does this by ranking the goal categories based on their shortest-path distance to the selected base categories. There are of course other options that have been considered in the literature. For example, Coursey and Mihalcea [17] proposed an alternative metric based on graph centrality, while Syed *et al.* [18] developed a spreading activation scheme to discover related concepts in a set of documents. Some of these ideas could be adapted into our method in future research.

However, even after settling on the shortest-path distance metric, there are many ways we could take into account the base categories' densities into the goal categories' ranking. The simplest option is to use it at a threshold value – to cut off base categories that have a density lower than a certain value, and then rank the goal categories according to which are closest to any remaining base category regardless of density. That is the approach we used in [4]. On the other hand, taking the density into account creates different conditions for the system. Since some base categories are now more important than others, it becomes acceptable, for example, to rank a goal that is further away from several high-density base categories higher than a goal that is

closer to a low-density base category. We thus define a ranking score for the goal categories, as the sum for all base categories of a ratio of their density to the distance separating the goal and base. There are several ways to compute this ratio; five options that we considered in this study are:

$$S_j = \sum_i D_i / (dist(i, j) + 0.0001) \quad (4)$$

$$S_j = \sum_i D_i / (dist(i, j)^2 + 0.0001) \quad (5)$$

$$S_j = \sum_i D_i e^{-dist(i,j)} \quad (6)$$

$$S_j = \sum_i D_i e^{-2\,dist(i,j)} \quad (7)$$

$$S_j = \sum_i D_i e^{-dist(i,j)^2} \quad (8)$$

In each of these equations, the score S_j of goal category j is computed as the sum, for all base categories i, of the density D_i of that category, which was computed in equation (3), divided by a function of the distance between categories i and j. This function is a simple division in equations (4) and (5), but the exponential in equations (6-8) put progressively more importance on the distance compared to the density. The addition of 0.0001 in equations (4) and (5) is simply to avoid a division by zero in the case where a selected base category is also a goal category.

Finally, the goal categories with the highest score are returned as classification results. In our current version of the system, we return the top three categories, to allow for queries to belong to several different categories. We believe that this corresponds to a human level of categorization; for example, in the KDD CUP 2005 competition [5], human labelers used on average 3.3 categories per query. However, this parameter is flexible, and we ran experiments keeping anywhere from one to five goal categories.

IV. Experimental Results

The various alternatives and options for our classifier described in the previous section were all implemented and tested, in order to study the behavior of the system and determine the optimal combination. That optimal combination was then subjected to a final set of tests with new data.

In order to compare and study the variations of our system, we submitted them all to the same challenge as the KDD CUP 2005 competition [5]. The 37 solutions entered in that competition were evaluated by classifying a set of 800 queries into up to five categories from a predefined set of 67 target categories c_j and comparing the results to the classification done by three human labelers. The solutions were ranked based on overall precision and overall F1 value, as computed by Equations (9-14). The competition's Performance Award was given to the system with the top overall F1 value, and the Precision Award was given to the system with the top overall precision value within the top 10 systems evaluated on overall F1 value. Overall Recall was not used in the competition, but is included here because it is useful in our experiments.

$$\text{Precision} = \frac{\sum_j \text{queries correctly labeled as } c_j}{\sum_j \text{queries labeled as } c_j} \quad (9)$$

$$\text{Recall} = \frac{\sum_j \text{queries correctly labeled as } c_j}{\sum_j \text{queries belonging to } c_j} \quad (10)$$

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (11)$$

$$\text{Overall Precision} = \frac{1}{3}\sum_{L=1}^{3} \text{Precision against labeler } L \quad (12)$$

$$\text{Overall Recall} = \frac{1}{3}\sum_{L=1}^{3} \text{Recall against labeler } L \quad (13)$$

$$\text{Overall F1} = \frac{1}{3}\sum_{L=1}^{3} \text{F1 against labeler } L \quad (14)$$

In order for our system to compare to the KDD CUP competition results, we need to use the same set of category labels. As we mentioned in Section 3, the size and level of detail of Wikipedia's category graph makes it possible to identify categories to map most sets of labels to. In our case, we identified 99 goal categories in Wikipedia corresponding to the 67 KDD CUP category set. These correspondences are presented in Appendix A.

A. Proportional Importance of Keywords

The first aspect of the system we studied is the different formulae for the proportional importance of query keywords in a title. As we explained in Section IIIB, the choice of formula has a direct impact on the system, as it determines which titles are more relevant given the user's query. This in turn determines the relevance of the base categories that lead to the goal categories. A bad choice at this stage can have an impact on the rest of the system.

The weight of a title, and of the query keywords it contains, is function of the two parameters presented in equation (1), namely the number of keywords present in the title and the importance of those keywords in that title. Section IIIB gives three possible mathematical definitions of keyword importance in a title. They are a straightforward proportion of keywords in the title, the proportion of characters in the title that belong to keywords, and the proportion of IDF of keywords to the total IDF of the title, as computed with equation (2). We implemented all three equations and tested the system independently using each. In all implementations, we limited the list of base categories to 25, weighted the goal categories using equation (5), and varied the number of returned goal categories from 1 to 5.

The results of these experiments are presented in Figure 2. The three different experiments are shown with different grey shades and markers: dark squares for the

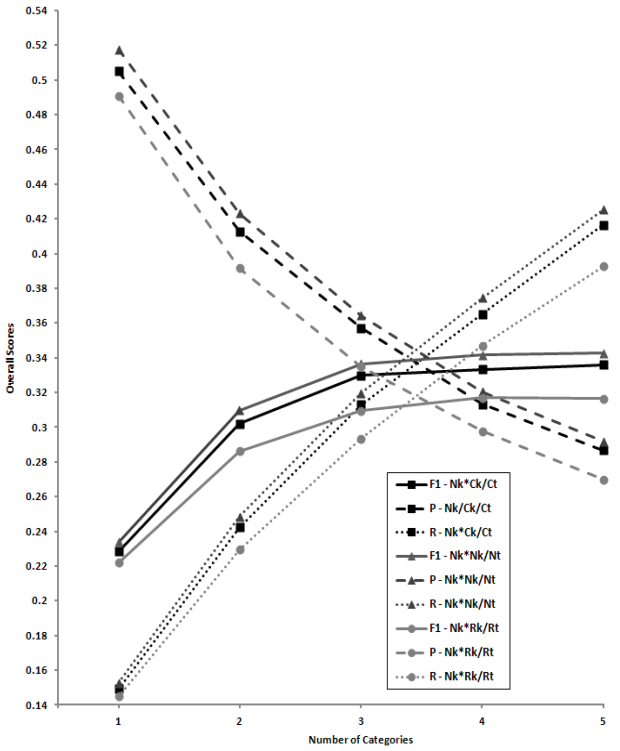

Figure 2. Overall precision (dashed line), recall (dotted line) and F1 (solid line) using $N_k*(C_k/C_t)$ (dark squares), $N_k*(N_k/N_t)$ (medium triangles), and $N_k*(\Sigma F_k/\Sigma F_t)$ (light circles).

formula using the proportion of characters, medium triangles for the formula using the proportion of words, and light circles for the formula using the proportion of IDF. Three results are also shown for each experiment: the overall precision computed using equation (12) in a dashed line, the overall recall of equation (13) in a dotted line, and the overall F1 of equation (14) in a solid line.

A few observations can be made from Figure 2. The first is that the overall result curves of all three variations have the same shape. This means that the system behaves in a very consistent way regardless of the exact formula used. There is no point where one of the results given one equation shoots off in a wildly different range of values from the other two equations. Moreover, while the exact difference in the results between the three equations varies, there is no point where they switch and one equation goes from giving worse results than another to giving better results. We can also see that the precision decreases and the recall increases as we increase the number of acceptable goal categories. This result was to be expected: increasing the number of categories returned in the results means that each query is classified in more categories, leading to more correct classification (that increase recall) and more incorrect classifications (that decrease precision). Finally, we can note that the best equation for the proportional importance of keywords in titles is consistently the proportion of keywords (N_k / N_t), followed closely by the proportion of characters (C_k / C_t), while the proportion of IDF (ΣF_k / ΣF_t) trails in third position.

TABLE I
COMPARISON OF IDF OF SAMPLE KEYWORDS

Keyword	T_w^*	F_w^*	T_w	F_w
WWE	2,705	7.8	657	9.0
Chief	83,977	5.6	1,695	8.1
Executive	82,976	5.8	867	8.7
Chairman	40,241	7.2	233	10.1
Headquartered	38,749	7.1	10	13.2

*Columns 2 and 3 are taken from [12].

It is surprising that the IDF measure gives the worst results of the three, when it worked well in other projects [12]. However, the IDF measure is based on a simple assumption, that a word with low semantic importance is one that is used commonly in most documents of the corpus. In our current system however, the "documents" are article titles, which are by design short, limited to important keywords, and stripped of semantically irrelevant words. These are clearly in contradiction with the assumptions that underlie the IDF measure. We can see this clearly when we compare the statistics of the keywords given in the example in [12] with the same keywords in our system, as we do in Table I. The system in [12] computed its statistics from the entire Wikipedia corpus, including article text, and thus computed reliable statistics; in the example in Table I the rarely-used company name WWE is found much more significant than the common corporate nouns chief, executive, chairman and headquartered. On the other hand, in our system WWE is used in almost as many titles as executive and has a comparable F_w score, which is dwarfed by the F_w score of chairman and headquartered, two common words that are very rarely used in article titles.

Finally, we can wonder if the two parts of equation (1) are really necessary, especially since the best equation we found for proportional importance repeats the N_k term. To explore that question, we ran the same test again using each part of the equation separately. Figure 3 plots the

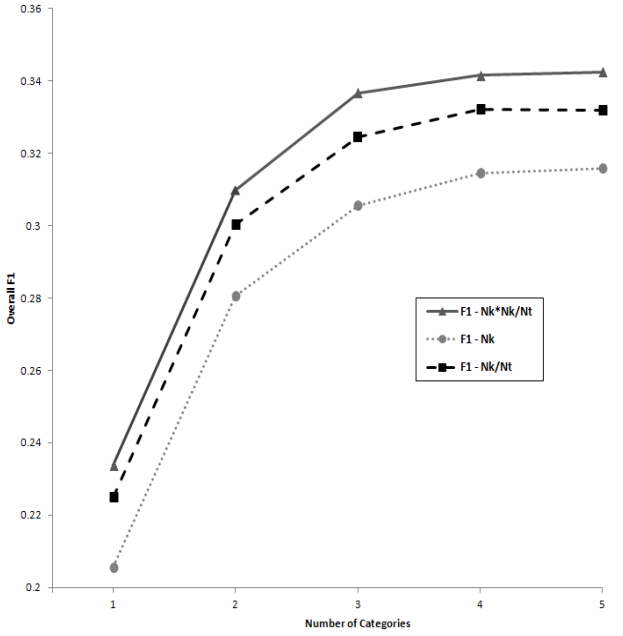

Figure 3. Overall F1 using $N_k*(N_k/N_t)$ (solid medium triangles), N_k/N_t (dashed dark rectangles), and N_k (light dotted circles).

overall F1 using N_k alone in light dotted line with circle markers, using N_k / N_t in black dashed line with square markers, and reproduces the overall F1 of N_k * (N_k / N_t) from Figure 3 in its medium solid line with triangle markers for comparison. This figure shows clearly that using the complete equation gives better results than using either one of its components.

B. Size of the Base Category Set

The second aspect of the system we studied comes at the end of the second stage of the algorithm, when the list of base categories is trimmed down to keep only the most relevant ones. This list will initially contain all categories connected to any title that contains at least one of the keywords the user specified. As we mentioned before, the average number of base categories generated by a query is 3,400 and the maximum is 45,000. These base categories are then used to compute the score of the goal categories, using one of the summations of equations (4-8). This test aims to see if the quality of the results can be improved by limiting the size of the set of base categories used in this summation, and if so what is the approximate ideal size.

For this test, we used $W_t = N_k$ * (N_k / N_t) for equation (1), the best formula found in the previous test. We again weighted the goal categories using equation (5) and varied the number of returned goal categories from 1 to 5. Figure 4 shows the F1 value of the system under these conditions when trimming the list of base categories to 500 (black solid line with diamonds), 100 (light solid line with circles), 50 (medium solid line with triangles), 25 (light dotted line with squares), 10 (black dashed line with squares) and 1 (black dotted line with circles).

Figure 4 shows clearly that the quality of the results drops if the set of base categories is too large (500) or too small (1). The difference in the results between the other four cases is less marked, and in fact the results with 10 and 100 base categories overlap. More notably, the results with 10 base categories start weaker than the case with 100, spike around 3 goal categories to outperform it, then drop again and tie it at 5 goal categories. This instability seems to indicate that 10 base categories are not enough. The tests with 25 and 50 base categories are the two that yield the best results; it thus seems then that the optimal size of the base category set is in that range. The 25 base category case outperforms the 50 case, and is the one we will prefer.

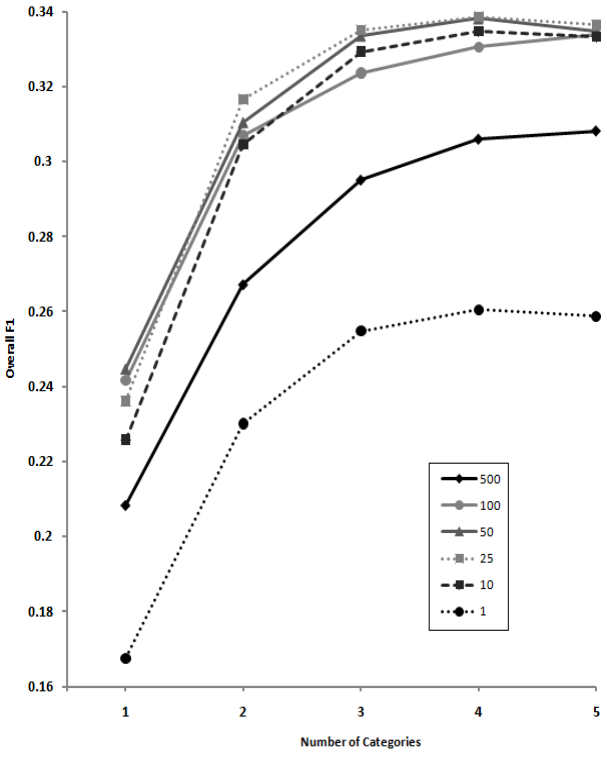

Figure 4. Overall F1 using from 1 to 500 base categories.

It is interesting to consider that in our previous study [4], we used the other alternative we proposed, namely to trim the set based on the density values. The cutoff we used was half the density value of the base category in the set with the highest density; any category with less than that density value was eliminated. This gave us a set of 28 base categories on average, a result which is consistent with the optimum we discovered in the present study.

C. Goal Category Score and Ranking

Another aspect of the system we wanted to study is the choice of equations we can use to account for the base categories' density and distance when ranking the goal categories. The option we used in the previous subsection, to find the nearest goals to any of the retained base categories regardless of their densities, is entirely valid. The alternative we consider here is to rank the goal categories in function of their distance to each base category and of the density of that base. We proposed five possible equations in Section IIIC to mathematically combine density and distance to rank the goal categories. Equation (4) considers both distance and density evenly, and the others put progressively more importance on the distance up to equation (8).

To illustrate the different impact of equations (4-8), consider three fictional base categories, one which has a density of 4 and is at a distance of 4 from a goal category, a second with a density of 4 and a distance of 3 from the same goal category, and the third with a density of 3 and a distance of 3 from the goal. The contribution of each of these bases to the goal category in each of the summations is given in Table II. As we can see in this table, the contribution of each base decreases as we move down from equation (4) to equation (8), but it decreases a lot more and a lot faster for the base at a distance of 4. The contribution to the summation of the category at a distance of 4 is almost equal to that of the categories at a distance of 3 when using equation (4), but is three orders

TABLE II
IMPACT OF THE GOAL CATEGORY EQUATIONS

Equation	Density 4 Distance 4	Density 4 Distance 3	Density 3 Distance 3
(4)	1.00	1.33	1.00
(5)	0.25	0.44	0.33
(6)	0.07	0.20	0.15
(7)	0.001	0.01	0.007
(8)	4.5×10^{-7}	0.0005	0.0004

of magnitude smaller when using equation (8). That is the result of putting more and more emphasis on distance rather than density: the impact of a farther-away higher-density category becomes negligible compared to a closer lower-density category. Meanwhile, comparing the contribution of the two categories of different densities at the same distance shows that, while they are in the same order of magnitude, the higher-density one is always more important than the lower-density one, as we would want.

We ran tests of our system using each of these five equations. In these tests, we again set $W_t = N_k * (N_k / N_t)$ for equation (1), kept the 25 highest-density base categories, and varied from retuning 1 to 5 categories. The overall F1 of the variations of the system is presented in Figure 5. In this figure, the classification results obtained using equation (4) are shown with a black dashed line with circle markers, equation (5) uses a grey solid line with square markers, equation (6) uses a dashed black line with triangle markers, equation (7) uses a light grey line with circle markers and equation (8) uses a grey line with triangle markers. For comparison, we also ran the classification using the exploration algorithm from our previous work [4], and included those results as a black dotted line with square markers.

We can see from Figure 5 that putting too much importance on distance rather than density can have a detrimental impact on the quality of the results: the results using equations (7) and (8) are the worst of the five equations. Even the results from equation (6) are of debatable quality: although it is in the same range as the results of equations (4) and (5), it shows a clear downward trend as we increase the number of goal categories considered, going from the best result for 2 goals to second-best with 3 goals to a narrow third place with 4 goals and finally to a more distant third place with 5 goals. Finally, we can see that the results using the exploration algorithm of [4] are clearly the worst ones, despite the system being updated to use the better category density equations and goal category mappings discovered in this study. The main difference between the two systems is thus the use of our old exploration algorithm to discover the goal categories nearest to any of the 25 base categories. This is also the source of the poorer results: the exploration algorithm is very sensitive to noise and outlier base categories that are near a goal category, and will return that goal category as a result. On the other hand, all five equations have in common that they sum the value of all base categories for each goal category, and therefore build-in noise tolerance. An outlier base category will seldom give enough of a score boost to one goal to eclipse the combined effect of the 24 other base categories on other goals.

Out of curiosity, we ran the same test a second time, but this time keeping the 100 highest-density base categories. These results are presented in Figure 6, using the same line conventions as Figure 5. It is interesting to see that this time it is equation (6) that yields the best results with a solid lead, not equation (5). This indicates a more fundamental relationship in our system: the best summation for the goal categories is not an absolute but depends on the number of base categories retained. With a smaller set of 25 base categories, the system works best when it considers a larger picture including the impact of more distant categories. But with a larger set of 100 base categories, the abundance of more distant categories seems to generate noise, and the system works best by limiting their impact and by focusing on closer base categories.

D. *Number of Goal Categories Returned*

The final parameter in our system is the number of goal categories to return. We have already explained in Section IIIC that our preference to return three goal categories is based on a study of human classification – namely, in the KDD CUP 2005 competition [5], human labelers used on average 3.3 categories per query. Moreover, looking at the F1 figures we presented in the previous subsections, we can see that the curve seems to be exponential, with each extra category returned giving a lesser increase in F1 value. Returning a fifth goal

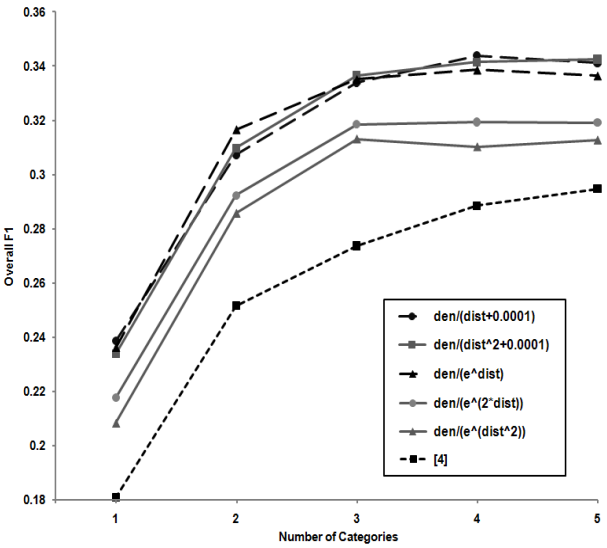

Figure 5. Goal score formulae using 25 base categories.

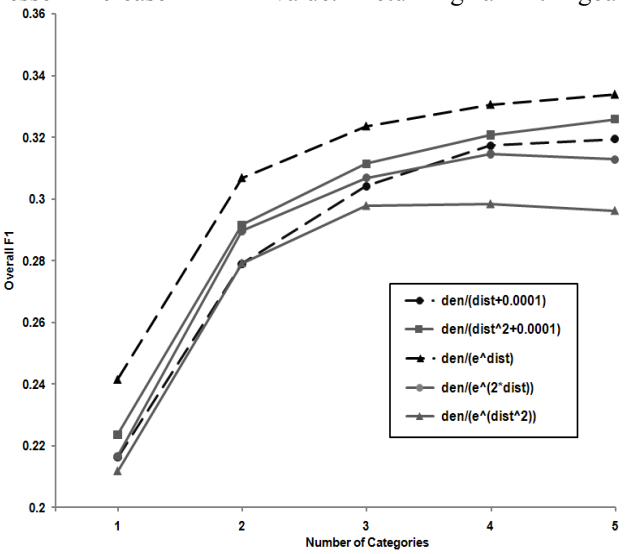

Figure 6. Goal score formulae using 100 base categories.

category gives the least improvement compared to returning only four goal categories, and in fact in some cases it causes a drop in F1. Returning three categories seems to be at the limit between the initial faster rate of increase of the curve and the later plateau.

Another way to look at the question is to consider the average score of goal categories at each rank, after summing the densities of the base categories for each and ranking them. If on average the top-ranked categories have a large difference to the rest of the graph, it will show that there exist a robust division between the likely-correct goal categories to return and the other goal categories. The opposite observation, on the other hand, would reveal that the rankings could be unstable and sensitive to noise, and that there is no solid score distinction between the goals our system returns and the others.

For this part of the study, we used the summation of equation (5). We can recall from our discussion in Section III that the maximum word weight is N_k and the maximum category density is N_k^2. Queries in the KDD CUP data set are at most 10 words long, giving a maximum base category density of 100. This in turn gives a maximum goal category score of 1,002,400 using equation (5) and 25 base categories in the case where the distance between the goal category and each of the base categories is one except for a single base category at a distance of zero; in other words, the goal is one of the base categories found and all other base categories are immediately connected to it. More realistically, we find in our experiments that the average base category density computed by equation (3) is 1.48, and the average distance between a base and goal category is 5.6 steps, so an average goal category score using equation (5) would be 1.18.

Figure 7 shows the average score of the goal category at each rank over all KDD CUP queries used in our experiment from the previous section, obtained using the method described above. This graph shows that the top category has on average a score of 3,272, several orders of magnitude above the average but still below our theoretical maximum. In fact, even the maximum score we observed in our experiments is only 67,506, very far below the theoretical maximum. This is due to the fact that most base categories are more than a single step removed from the goal category.

The graph also shows the massive difference between the first three ranks of goal categories and the other 96. The average score goes from 3,272, to 657 at rank 2 and 127 at rank 3, down to 20 and 16 at ranks 4 and 5 respectively, then cover the interval from 2 to 0.7 between ranks 6 and 99. This demonstrates a number of facts. First of all, both the values of the first five goal ranks and the differences between their scores when compared to the other 94 shows that these first ranks are resilient to noise and variations. It also justifies our decision to study the performance of our system using the top 1 to 5 goal categories, and it gives further experimental support to our decision to limit the number of goal categories returned by the classifier to three.

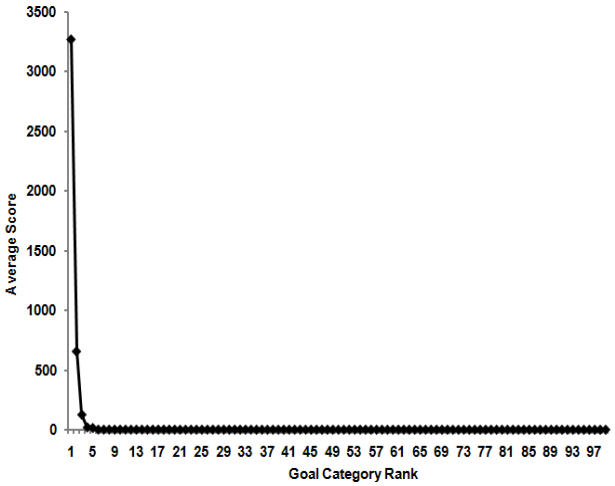

Figure 7. Average goal category score per rank over all KDD CUP queries.

It is interesting to note that the average score of the categories over the entire distribution is 42.53, very far off from our theoretical average of 1.18. However, if we ignore the first three ranks, whose values are very high outliners in this distribution, the average score becomes 1.62. Moreover, the average score over ranks 6 to 99 is 1.28. Both of these values are in line with the average we expected to find.

E. The Optimal System

After having performed these experiments, we are ready to put forward the optimal classifier, or the one that combines the best features from the options we have studied. This classifier uses $W_t = N_k * (N_k / N_t)$ for equation (1), selects the top 25 base categories, ranks the goal categories using the summation formula of equation (5), and returns the top-three categories ranked. The results we obtain with that system are presented in Table III, with other KDD CUP competition finalists reported in [5] for comparison. Note that participants had the option to enter their system for precision ranking but not F1 ranking or vice-versa rather than both precision and F1 ranking, and several participants chose to use that option. Consequently, there are some N/A values in the results in Table III. As can be seen from Table III, our system performs well above the competition average, and in fact ranks in the top-10 of the competition in F1 and in the top-5 in precision. For comparison, system #22, which

TABLE III
CLASSIFICATION RESULTS

System	F1 Rank	Overall F1	Precision Rank	Overall Precision
KDDCUP #22	1	0.4444	N/A	0.4141
KDDCUP #37	N/A	0.4261	1	0.4237
KDDCUP #21	6	0.3401	2*	0.3409
Our system	**7**	**0.3366**	**2**	**0.3643**
KDDCUP #14	7*	0.3129	N/A	0.3173
Our previous work	10	0.2827	7	0.3065
KDDCUP Mean		0.2353		0.2545
KDDCUP Median		0.2327		0.2446

* indicates competition systems that would have been outranked by ours.

TABLE IV
SAMPLE BASE CATEGORIES

Category	Rank	Density	Titles
Internet Explorer	1	4	36
Internet history	2	4	32
Windows web browsers	3	4	20
Microsoft criticisms and controversies	8	4	4
HTTP	25	2.67	5
Mobile phone web browsers	26	2.67	4
Cascading Style Sheets	33	2	5
Internet	37	1	17
PlayStation Games	660	0.4	2
Islands of Finland	905	0.33	1
History of animation	1811	0.05	1

TABLE V
SAMPLE GOAL CATEGORIES

Goal Category	Rank	Score
Internet	1	11.51
Software	2	10.09
Computing	3	8.03
Internet culture	4	5.63
Websites	5	4.97
Technology	16	3.54
Magazines	18	3.39
Industries	30	2.86
Law	49	2.54
Renting	99	1.30

Refer to Appendix A for the list of KDDCUP categories corresponding to these goal categories.

won the first place in the competition, achieved the best results by querying multiple web search engines and aggregating their results [2]. Our method may not perform as well right now, but it offers the potential for algorithmic and knowledge-base improvements that goes well beyond those of a simple aggregate function, and is not dependent on third-party commercial technology.

We also updated the the classifier we built in our previous work of [4] to use $W_t = N_k * (N_k / N_t)$, the top-25 base category cutoff and the goal category mapping of Appendix A. Its original iterative graph exploration method was also slightly modified to explore all paths rather than parents-only, to break ties using equation (5) rather than a random draw, and to return the top-three goal categories rather than the top-five. These modifications are all meant to update the system of [4] with the best features obtained in this research, to create a fair comparison. The results we obtained are included in Table III. While it does perform better than the average KDDCUP system, we find that our previous classifier still falls short of the one we studied in this paper.

We also found in our results that 47 of the 800 test queries were not classified at all, because the algorithm failed to select any base categories at all. This situation occurs when no Wikipedia titles featuring query words can be found. These queries are all single words, and that word is either an uncommon abbreviation (the query "AATFCU" for example), misspelled in an unusual way ("egyptains"), an erroneous compounding of two words ("contactlens"), a rare website URL, or even a combination of the above (such as the misspelled URL "studioeonline.com" instead of "studioweonline.com"). These are all situations that occur with real user search queries, and are therefore present in the KDDCUP data set. It is worth noting that Wikipedia titles include common cases of all these errors, so that only the 5.9% most unusual cases lead to failure in our system.

It could be interesting to study a specific example, to see the system's behavior step by step. We chose for this purpose to study a query for "internet explorer" in the KDDCUP set. This query was manually classified by the competition's three labelers, into the KDDCUP categories "Computers\Software; Computers\Internet & Intranet; Computers\Security; Computers\Multimedia; Information\Companies & Industries" by the first labeler, into "Computers\Internet & Intranet; Computers\Software" by the second labeler, and into "Computers\Software; Computers\Internet & Intranet; Information\Companies & Industries" by the third labeler.

The algorithm begins by identifying a set of relevant base categories using the procedure explained in Section IIIB and then weighting them using equation (3). For this query, our algorithm identifies 1,810 base categories, and keeps the 25 highest-density ones, breaking the tie for number 25 by considering the number of titles pointing to the categories as we explained in Section IIIB.

For any two-word query, the maximum title weight value that can be computed by equation (1) is 2, and the maximum base category density value that can be returned by equation (3) is 4. And in fact, we find that 8 categories receive this maximum density, including some examples we listed in Table IV. We can see from these examples that the top-ranked base categories are indeed very relevant to the query. Examining the entire set of base categories reveals that the density values drop to half the maximum by rank 33, and to a quarter of it by rank 37. The density value continues to drop as we go down the list: the average density of a base category in this example is 0.4 which corresponds to rank 660, by the middle of the list at rank 905 the density is 0.33, and the final category in the list has a density of only 0.05. It can also be seen from the samples in Table IV that the relevance of the categories to the query does seem to decrease along with the density value. Looking at the complete list of 1,810 base categories, we find that the first non-software-related category is "Exploration" at rank 41 with a density of 1. But software-related categories continue to dominate the list, mixed with a growing number of non-software categories, until rank 354 (density of 0.5 and 1 title pointing to the category) where non-computer categories begin to dominate. Incidentally, the last software-related category in the list is "United States internet case law", at rank 1791 with a density of 0.11.

The next step of our algorithm is to rank the 99 goal categories using the sum of density values in equation (5). Sample rankings are given in Table V. This table uses the Wikipedia goal category labels; the matching KDDCUP categories can be found in Appendix A. We can see from these results that the scores drop by half from the first

TABLE VI
TEST CLASSIFICATION RESULTS

Query set	Overall F1	Overall Precision	Overall Recall
KDDCUP 111	0.3636	0.4254	0.3175
TREC	0.4639	0.4223	0.5267
KDDCUP 800	0.3366	0.3643	0.3195

result to the fourth one. This is much less drastic than the drop we observed on average in Figure 7, but is nonetheless consistent as it shows a quick drop from a peak over the first three ranks and a long and more stable tail over ranks 4 to 99.

It is also encouraging to see that the best two goal categories selected by our system correspond to "Computers\Internet & Intranet" and "Computers\Software", the only two categories to be picked by all three KDDCUP labelers. The fourth goal corresponds to "Online Community/Other" and is the first goal that is not in the KDDCUP "Computer/" category, although it is still strongly relevant to the query. Further down, the first goal that corresponds neither to a "Computers/" nor "Online Community/" category is Technology ("Information\Science & Technology") at rank 16, which is still somewhat related to the query, and the first truly irrelevant result is Magazines ("Living\Book & Magazine") at rank 18 with a little over a quarter of the top category's score. Of the categories picked by labelers, the one that ranked worst in our system was "Information\Companies & Industries" at rank 30. All the other categories they identified are found in the top-10 results of our system.

F. New Data and Final Tests

In order to show that our results in Table III are general and not due to picking the best system for a specific data set, we ran two more tests of our system with two new data sets.

The first data set is a set of 111 KDD CUP 2005 queries classified by a competition judge. This set was not part of the 800 test queries we used previously; it was a set of queries made available by the competition organizers to participants prior to the competition, to develop and test their systems. Naturally, the queries in this set will be similar to the other KDD CUP queries, and so we expect similar results.

The second data set is a set of queries taken from the TREC 2007 Question-Answering (QA) track [19]. That data set is composed of 445 questions on 70 different topics; we randomly selected three questions per topic to use for our test. It is also worth noting that the questions in TREC 2007 were designed to be asked sequentially, meaning that a system could rely on information from the previous questions, while our system is designed to classify each query by itself with no query history. Consequently, questions that were too vague to be understood without previous information were disambiguated by adding the topic label. For example, the question 'Who is the CEO?' in the series of questions on the company 3M was rephrased as 'Who is the CEO of 3M?' Finally, two of the co-authors independently labeled the questions to KDD CUP categories in order to have a standard to compare our system's results to in Equations (9) and (10). The TREC data set was selected in order to subject our system to very different testing conditions: instead of the short keyword-only KDD CUP web queries, TREC has long and grammatically-correct English questions.

TABLE VII
CATEGORIZATION AND RECALL

Query set	Average number of categories	Recall
TREC Labeler 1	1.93 ± 0.81	0.5443
TREC Labeler 2	2.91 ± 0.92	0.5090
KDDCUP Labeler 2	2.39 ± 0.93	0.3763
KDDCUP Labeler 1	3.67 ± 1.13	0.3076
KDDCUP Labeler 3	3.85 ± 1.09	0.2747

The results from both tests are presented in Table VI, along with our system's development results already presented in Table III for comparison. These results show that our classifier works better with the test data than with the training data it was developed and optimized on. This counter-intuitive result requires explanation.

The greatest difference in our results is on recall, which increases by over 20% from the training KDDCUP test to the TREC test. Recall, as presented in equation (10), is the ratio of correct category labels identified by our system for a query to the total number of category labels the query really has. Since our classifier returns a fixed number of three categories per query, it stands to reason that it cannot achieve perfect recall for a query set that assigns more than three categories, and that it can get better recall on a query set that assigns fewer categories per query. To examine this hypothesis, we compared the results of five of our labelers individually: the three labelers of the KDDCUP competition and the two labelers of the TREC competition (the 111 KDDCUP demo queries, having been labeled by only one person, were not useful for this test). Specifically, we looked at the average number of categories per query each labeler used and the recall value our system achieved using that query set. The results, presented in Table VII, show that our intuition is correct: query sets with less categories per query lead to higher recall, with the most drastic example being the increase of 1.5 categories per query between KDDCUP labelers 2 and 3 that yielded a 10% decrease in recall. However, it also appears from that table that the relationship does not hold across different query sets: KDDCUP labeler 2 assigns less labels per query that TREC labeler 2 but still has a much lower recall.

Next, we can contrast the two KDDCUP tests: they both had nearly identical recall but the new data gave a 6% increase in precision. This is interesting because the queries are from the same data sets, they are web keyword searches of the same average length, and the correct categorization statistics are nearly identical to those of Labeler 3 so we would actually expect the recall to be lower than it ended up being. An increase in both precision and recall can have the same origin in equations

(9) and (10): a greater proportion of correct categories identified by our classifier. But everything else being equal, this would only happen if the queries themselves were easier for our system to understand. To verify this hypothesis, we checked both query sets for words that are unknown in our system. As we explained previously, a lot of these words may be rare but simple typos ("egyptains") or missing spaces between two words ("contactlens"), and while they are unknown and ignored in our system their meaning is immediately obvious to the human labelers. The labelers thus have more information to classify the queries, which makes it inherently more difficult for our system to generate the same classification. Upon evaluation of our data, we find that the KDDCUP set of 800 queries features about twice the frequency of unknown words of the set of 111 queries. Indeed, 10.4% of queries in the 800-query set have unknown words and 4.4% of words overall are unknown, while only 5.4% of queries in the 111-query set have unknown words and only 2.5% of words in that set are unknown. This is an important difference between the two query sets, and we believe it explains why the 111 queries are more often classified correctly. It incidentally also indicates that an automated corrector should be incorporated in the system in the future.

The better performance of our system on the TREC query set can be explained in the same way. Thanks to the fact that set is composed of correct English questions, it features even fewer unknown words: a mere 0.4% of words in 1.9% of queries. Moreover, for the same reason, the queries are much longer: on average 5.3 words in length after stopword removal, compared to 2.4 words for the KDDCUP queries. This means that even if there is an unknown word in a query, there are still a lot of other words in the TREC queries for our system to make a reasonably good classification.

Differences in the queries aside, it does not appear to be major distinctions, much less setbacks, when using our classifier on new and unseen data sets. It seems robust enough to handle new queries in a different spread of domains, and to handle both web-style keyword searches and English questions without loss of precision or recall.

Finally, it could be interesting to determine how our classifier's performance compares to that of a human doing the same labeling task. Query classification is a subjective task: since queries are short and often ambiguous, their exact meaning and classification is often dependent on human interpretation [20]. It is clear from Table VII that this is the case for our query sets, that human labelers do not agree with each other on the classification of these queries. We can evaluate the human labelers by computing the F1 of each one's classification compared to the others in the same data set. In the case of the KDDCUP data, the average F1 of human labelers is known to be between 0.4771 and 0.5377 [5], while for our labeled TREC data we can compute the F1 between the two human labelers to be 0.5605. This means our system has between 63% and 71% of a human's performance when labeling the KDDCUP queries, and 83% of a human's performance when labeling the TREC queries. It thus appears that by this benchmark, our classifier again performs better on the TREC data set than on the KDDCUP one. This gives further weight to our conclusion that our system is robust enough to handle very diverse queries.

V. Conclusion

In this paper, we presented a ranking and classification algorithm to exploit the Wikipedia category graph to find the best set of goal categories given user-specified keywords. To demonstrate its efficiency, we implemented a query classification system using our algorithm. We performed a thorough study of the algorithm in this paper, focusing on each design decision individually and considering the practical impact of different alternatives. We showed that our system's classification results compare favorably to those of the KDD CUP 2005 competition: it would have ranked 2nd on precision with a performance 10% better than the competition mean, and 7th in the competition on F1. We further detailed the results of an example query in key steps of the algorithm, to demonstrate that each partial result is correct. And finally we presented two blind tests on different data sets that were not used to develop the system, to validate our results.

We believe this work will be of interest to anyone developing query classification systems, text classification systems, or most other kinds of classification software. By using Wikipedia, a classification system gains the ability to classify queries into a set of almost 300,000 categories covering most of human knowledge and which can easily be mapped to a simpler application-specific set of categories when needed, as we did in this study. And while we considered and tested multiple alternatives at every design stage of our system, it is possible to conceive of further alternatives that could be implemented on the same framework and compared to our results. Future work can focus on exploring these alternatives and further improving the quality of the classification. In that respect, as we indicated in Section IV.F, one of the first directions to work in will be to integrate an automated corrector into the system, to address the problem of unknown words.

Appendix A

This appendix lists how we mapped the 67 KDD CUP categories to 99 corresponding Wikipedia categories in the September 2008 version of the encyclopedia.

KDD CUP Category	Wikipedia Category
Computers\Hardware	Computer hardware
Computers\Internet & Intranet	Internet
	Computer networks
Computers\Mobile Computing	Mobile computers
Computers\Multimedia	Multimedia
Computers\Networks & Telecommunication	Networks
	Telecommunications

Computers\Security	Computer security
Computers\Software	Software
Computers\Other	Computing
Entertainment\Celebrities	Celebrities
Entertainment\Games & Toys	Games
	Toys
Entertainment\Humor & Fun	Humor
Entertainment\Movies	Film
Entertainment\Music	Music
Entertainment\Pictures & Photos	Photographs
Entertainment\Radio	Radio
Entertainment\TV	Television
Entertainment\Other	Entertainment
Information\Arts & Humanities	Arts
	Humanities
Information\Companies & Industries	Companies
	Industries
Information\Science & Technology	Science
	Technology
Information\Education	Education
Information\Law & Politics	Law
	Politics
Information\Local & Regional	Regions
	Municipalities
	Local government
Information\References & Libraries	Reference
	Libraries
Information\Other	Information
Living\Book & Magazine	Books
	Magazines
Living\Car & Garage	Automobiles
	Garages
Living\Career & Jobs	Employment
Living\Dating & Relationships	Dating
	Intimate relationships
Living\Family & Kids	Family
	Children
Living\Fashion & Apparel	Fashion
	Clothing
Living\Finance & Investment	Finance
	Investment
Living\Food & Cooking	Food and drink
	Cooking
Living\Furnishing & Houseware	Decorative arts
	Furnishings
	Home appliances
Living\Gifts & Collectables	Giving
	Collecting
Living\Health & Fitness	Health
	Exercise
Living\Landscaping & Gardening	Landscape
	Gardening
Living\Pets & Animals	Pets
	Animals
Living\Real Estate	Real estate
Living\Religion & Belief	Religion
	Belief
Living\Tools & Hardware	Tools
	Hardware (mechanical)
Living\Travel & Vacation	Travel
	Holidays
Living\Other	Personal life
Online Community\Chat & Instant Messaging	On-line chat
	Instant messaging
Online Community\Forums & Groups	Internet forums
Online Community\Homepages	Websites
Online Community\People Search	Internet personalities
Online Community\Personal Services	Online social networking
Online Community\Other	Virtual communities
	Internet culture
Shopping\Auctions & Bids	Auctions and trading
Shopping\Stores & Products	Retail
	Product management
Shopping\Buying Guides & Researching	Consumer behaviour
	Consumer protection
Shopping\Lease & Rent	Renting
Shopping\Bargains & Discounts	Sales promotion
	Bargaining theory
Shopping\Other	Distribution, retailing, and wholesaling
Sports\American Football	American football
Sports\Auto Racing	Auto racing
Sports\Baseball	Baseball
Sports\Basketball	Basketball
Sports\Hockey	Hockey
Sports\News & Scores	Sports media
Sports\Schedules & Tickets	Sport events
	Seasons
Sports\Soccer	Football (soccer)
Sports\Tennis	Tennis
Sports\Olympic Games	Olympics
Sports\Outdoor Recreations	Outdoor recreation
Sports\Other	Sports

REFERENCES

[1] Maj. B. J. Jansen, A. Spink, T. Saracevic, "Real life, real users, and real needs: a study and analysis of user queries on the web", *Information Processing and Management*, vol. 36, issue 2, 2000, pp. 207-227.

[2] D. Shen, R. Pan, J.-T. Sun, J. J. Pan, K. Wu, J. Yin, Q. Yang, "Q2C@UST: our winning solution to query classification in KDDCUP 2005", *ACM SIGKDD Explorations Newsletter*, vol. 7, issue 2, 2005, pp. 100-110.

[3] M. Alemzadeh, F. Karray, "An efficient method for tagging a query with category labels using Wikipedia towards enhancing search engine results", *2010 IEEE/WIC/ACM International Conference on Web*

Intelligence and Intelligent Agent Technology, Toronto, Canada, 2010, pp. 192-195.

[4] M. Alemzadeh, R. Khoury, F. Karray, "Exploring Wikipedia's Category Graph for Query Classification", in *Autonomous and Intelligent Systems*, M. Kamel, F. Farray, W. Gueaieb, A. Khamis (eds.), Lecture notes in Artificial Intelligence, 1st edition, vol. 6752, Springer, 2011, pp. 222-230.

[5] Y. Li, Z. Zheng, H. Dai, "KDD CUP-2005 report: Facing a great challenge", *ACM SIGKDD Explorations Newsletter*, vol. 7 issue 2, 2005, pp. 91-99.

[6] D. Shen, J. Sun, Q. Yang, Z. Chen, "Building bridges for web query classification", *Proceedings of SIGIR'06*, 2006, pp. 131-138.

[7] H. Cao, D. H. Hu, D. Shen, D. Jiang, J.-T. Sun, E. Chen, and Q. Yang. "Context-aware query classification", *Proceedings of SIGIR*, 2009.

[8] J. Fu, J. Xu, K. Jia, "Domain ontology based automatic question answering", *International Conference on Computer Engineering and Technology (ICCET '08)*, vol. 2, 2009, pp. 346-349.

[9] J. Yu, N. Ye, "Automatic web query classification using large unlabeled web pages", *9th International Conference on Web-Age Information Management*, Zhangjiajie, China, 2008, pp. 211-215.

[10] S. M. Beitzel, E. C. Jensen, D. D. Lewis, A. Chowdhury, O. Frieder, "Automatic classification of web queries using very large unlabeled query logs", *ACM Transactions on Information Systems*, vol. 25, no. 2, 2007, article 9.

[11] J. Hu, G. Wang, F. Lochovsky, J.-T. Sun, Z. Chen, "Understanding user's query intent with Wikipedia", *Proceedings of the 18th international conference on World Wide Web*, Spain, 2009, pp. 471-480.

[12] R. Khoury, "Query Classification using Wikipedia", *International Journal of Intelligent Information and Database Systems*, vol. 5, no. 2, April 2011, pp. 143-163.

[13] P. Schönhofen, "Identifying document topics using the Wikipedia category network", *Web Intelligence and Agent Systems*, IOS Press, Vol. 7, No. 2, 2009, pp. 195-207.

[14] Z. Minier, Z. Bodo, L. Csato, "Wikipedia-based kernels for text categorization", *International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, Romania, 2007, pp, 157-164.

[15] P. Wang, J. Hu, H.-J. Zeng, Z. Chen, "Using Wikipedia knowledge to improve text classification", *Knowledge and Information Systems*, vol. 19, issue 3, 2009, pp. 265-281.

[16] S. Banerjee, K. Ramanathan, A. Gupta, "Clustering short texts using Wikipedia," *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, Amsterdam, Netherlands, 2007 pp. 787-788.

[17] K. Coursey, R. Mihalcea, "Topic identification using Wikipedia graph centrality", *Proceedings of NAACL HLT*, 2009, pp. 117-120.

[18] Z. S. Syed, T. Finin, A. Joshi, "Wikipedia as an ontology for describing documents", *Proceedings of the Second International Conference on Weblogs and Social Media*, March 2008.

[19] H. T. Dang, D. Kelly, J. Lin, "Overview of the TREC 2007 question answering track", *Proceedings of the Sixteenth Text Retrieval Conference*, 2007.

[20] B. Cao, J.-T. Sun, E. W. Xiang, D. H. Hu, Q. Yang, Z. Chen, "PQC: personalized query classification", *Proceedings of the 18th ACM conference on information and knowledge management*, Hong Kong, China, 2009, pp. 1217-1226.