Using Encyclopaedic Knowledge for Query Classification

Richard Khoury

1Department of Software Engineering, Lakehead University, Thunder Bay, Ontario, Canada

Abstract - Identifying the intended topic that underlies a user’s query can benefit a large range of applications, from search engines to question-answering systems. However, query classification remains a difficult challenge due to the variety of queries a user can ask, the wide range of topics users can ask about, and the limited amount of information that can be mined from the query. In this paper, we develop a new query classification system that accounts for these three challenges. Our system relies on encyclopaedic knowledge to understand the user’s query and fill in the gaps of missing information. Specifically, we use the freely-available online encyclopaedia Wikipedia as a natural-language knowledge base, and exploit Wikipedia’s structure to infer the correct classification of any user query.

Keywords: Query classification; Natural language processing; Knowledge discovery; Knowledge representation;

1 Introduction

Query classification is the branch of Natural Language Processing (NLP) whose goal is to identify the category label, in a predefined set, that best represents the domain of the question being asked. While similar categorization tasks are found in several branches of NLP, query classification is a uniquely difficult challenge due to the fact that a typical query is only a few words long and includes an implicit subjective domain intended by the user [1]. Moreover, queries can come in two very different styles: either as complete and grammatically-correct questions the kind a person would ask to a question-answering system or another human being, or as web-style keyword-only searches. Finally, given the growing shift in focus of the NLP community towards web queries, general classification systems now have to handle queries dealing with any domain found on the internet, which is to say any domain at all [2].

In this paper, we present a new query classification system which addresses all of the challenges described above by using encyclopaedic knowledge. We decided to build our system based on the Wikipedia database, a design decision which brings a massive amount of natural language data from which to infer the user’s implicit intent, as well as a set of approximately 300,000 categories covering most domains of human knowledge. In experimental results, our proposed system matches top classification systems found in the literature and handles both web-style keyword queries and grammatically-complete English questions.

The rest of the paper is organized as follows. Section 2 presents overviews of the literature in relevant fields, namely in query classification and on the more general topic of NLP research using Wikipedia. We present in detail our classification system in Section 3, then we move on in Section 4 to describe two sets of experiments performed on this system and to analyse their results. Finally, we give some concluding remarks in Section 5.

2 Related Work

Query classification is the branch of NLP that focuses on inferring the domain of user-written queries, and on assigning each query to the category label that best represents its domain in a predefined set of labels. The main challenge in that field comes from the scarcity of information in a typical query. Indeed, while other NLP systems can use information ranging from multi-word windows to entire documents, the typical query is very short. Jansen et al. [2] have shown that 62% of queries feature two terms or less, while 79% of queries used in the KDD CUP 2005 competition featured four terms or less [3]. On the other hand, users’ queries might be about any topic at all. A query classification system thus faces the challenge of pinpointing practically any domain based on only two to four keywords on average.

Given the sparsity of information that can be gathered directly from a query, many researchers opt to design query classification systems that are enriched by an outside knowledge source. One popular source, when available, is a domain ontology. For example, Fu et al. [4] use a music knowledge ontology of 54 classes in their research. However, when such ontologies do not already exist, researchers have had to create their own knowledge base from various sources. Researchers have proven to be quite imaginative and resourceful in this regard. For instance, Beitzel et al. [5] use a database of 20,000 manually-classified web queries as a knowledge base. In an alternative approach, Shen et al. [3] rely on a knowledge base automatically constructed from a corpus of web pages categorised by a commercial search engine. Jingbo and Na [6] go a step further, and automatically
classify their web page corpus using on some domain-specific seed keywords they specify. One last interesting sample of this type of project is that of Hu et al. [7], who built a graph out of a subset of interconnected Wikipedia categories and articles related to some seed concepts they selected. Next, they traverse the graph using a Markov random walk algorithm to assign a probability to each connection. A user query in a related concept can then be classified to the most probable article or category in the graph.

The idea of using an external knowledge base to enrich a system is common to many areas of NLP. Moreover, while it seems that only Hu et al. experimented with Wikipedia for query classification [7], that resource has been used successfully in other NLP projects. For example, Schönhofen [8] developed a document classification system that uses Wikipedia’s categories as class labels. Schönhofen’s document classification system finds the list of Wikipedia titles whose words appear in the document’s text, matches them to their corresponding articles, gets the list of categories from these articles, and weights them based on a number of factors. Ahn et al. [9] designed a question-answering (QA) system which finds the answer in Wikipedia. Their system searches Wikipedia for an article related to the question, then scans it for named entities and weights each entity’s relative importance to the article and the query. Wee and Hassan [10] proposed an algorithm to compute the similarity between two words, based simply on the ratio of the number of Wikipedia articles in which both appeared to the number of articles in which only one appeared. In experiments, all three of these systems were found to perform better than the literature benchmarks.

The main advantage of using Wikipedia as a knowledge base is its sheer size [8], [9], [10]. Many NLP applications require access to a large knowledge base, and while a great number of encyclopaedias, both general and domain-specific, are available to researchers today to help them in these projects, Wikipedia is orders of magnitude larger than all but a few extremely specialized resources [11], [12]. In fact, Wee and Hassan explicitly credited the good performance of their word similarity system in part to the fact that Wikipedia contains a much larger and more diverse vocabulary than other resources used for similar applications [10]. For the same reason, we opted to use Wikipedia as a basis for our research as well.

3 Methodology

3.1 Corpus preparation

Before we can work efficiently with the Wikipedia corpus, we have to make it undergo a set of preparation steps. These steps are meant to remove unnecessary information and to organize the retained information in a structured database.

Each Wikipedia page has a unique name, its title. Our first processing step is to perform stopword removal and stemming on the titles. We then insure that each title points only to real articles, by following the redirect links and expanding the disambiguation pages that are commonly found in Wikipedia. In the second step, we consider each individual article. We delete those that are not encyclopaedic topics, such as Wikipedia’s discussion pages. Next, we separate the list of categories present at the end of the article from the rest of the article text. The text is trimmed to retain only those words that were inside the wikilinks’ double-square-bracket mark-up code. Indeed, according to Wikipedia’s policy, only the most important notions in an article should be wikilinked to the relevant related article. Through this processing step, we thus reduce an article text to its set of most important keywords. The final processing step consists in filtering out categories that are useless for query classification, namely those meant for Wikipedia administration, and in merging “stub” categories with their matching real categories.

3.2 Query Classification

The aim of our query classification algorithm is to assign any user’s query to the most significant category in the set extracted from Wikipedia in the corpus preparation stage. As we explained in Sections 1 and 2, the size of the Wikipedia corpus should allow our system to recognize queries on practically any subject. Our classification algorithm is designed to exploit the structure of our prepared corpus in a step-by-step manner, going from the query words to titles, articles, and finally categories.

In the first step of our algorithm, we begin by submitting the user’s query to stopword removal and stemming, and we filter out words in the query that are not part of our corpus. Next, we assign a weight $R_w$ to each word $w$ in the query:

$$R_w = \frac{1}{3} \left[ \ln \left( \frac{N_t}{W_t} \right) + \ln \left( \frac{N_a}{W_a} \right) + \ln \left( \frac{N_c}{W_c} \right) \right]$$

(1)

In Equation (1), $N_t$, $N_a$, and $N_c$ are respectively the number of titles, articles and categories in our corpus, while $W_t$, $W_a$, and $W_c$ are respectively the number of titles, articles and category vocabularies featuring word $w$. We defined a category vocabulary as the words of the titles of all articles pointing to it, excluding redirect titles and disambiguation titles.

The next step is to gather the set of titles that feature at least one of the query words. We compute the weight $R_t$,
of each title \( t \) as a sum of the weights of the query words it features:

\[
R_t = \sum_w \frac{R_a \times f(w,t)}{L_Q}
\]  

(2)

In Equation (2), \( L_Q \) is the length in number of words of the user’s query, and \( f(w,t) \) is a binary function defined as:

\[
f(w,t) = \begin{cases} 
1 & \text{if } w \text{ occurs in } t \\
0 & \text{if } w \text{ does not occur in } t
\end{cases}
\]  

(3)

In the third step, we follow the links from each of the titles we selected to every article it points to and generate an exhaustive list of title-article pairs. We then filter these pairs using the criterion that the relevant pairs should feature most or all of the query words. Mathematically, we impose that the title-article pair should feature all query words if the query is four words long or less, and all-but-the-least-significant-word if the query has five words or more, where the least significant word is simply the one with the lowest \( R_a \) weight. If no title-article pairs are retained at this step, we iteratively remove the least significant word and try again, until results are obtained.

The fourth step assigns a weight \( R_a \) to each article \( a \). This weight is simply the maximum weight from all the titles that point to it, as shown in Equation (4). We also experimented with summing the weights instead of taking the maximum, but we found this to be an unreliable measure because the number of titles pointing to an article is not a metric of the article’s importance or relevance, but simply an artefact of Wikipedia’s structure. In that respect, our observation echoes a similar conclusion reached by Schönhofen [8].

\[
R_a = \max_t R_t
\]  

(4)

The fifth step is to compute the weight \( R_c \) of each category \( c \) pointed to by the articles. This category weight is defined as the sum of the weight of each article pointing to it as in Equation (5). The reason we use a sum rather than a maximum as we did in Equation (4) is that the number of different articles is meaningful. Indeed, following our corpus preparation stage, each article represents a distinct encyclopaedic topic. Consequently, the number of articles pointing to a category represents how many different topics significantly relate to that category subject. Finally, the categories are ranked by \( R_c \) weight, and the category with the maximum weight is returned as the classification result.

\[
R_c = \sum_a R_a
\]  

(5)

4 Experimental Results

In order to thoroughly test our query classification system, we ran two independent sets of tests. These tests are meant to reflect the two main types of questions that can be asked by users, namely grammatically-complete and correct English questions and keyword-based web queries. We used standard and publicly-available question corpora for each of these tests. For the keyword-based test we used the KDD CUP 2005 corpus of web queries, and for the test on grammatically-complete questions we used the set of questions from the TREC 2007 QA track.

4.1 KDD CUP 2005

KDD CUP is the ACM’s annual Data Mining and Knowledge Discovery competition. In 2005, the topic of the competition was “Internet user search query categorization”, and a complete technical report on the competition and its outcome was prepared by Li et al. [1]. Participants in the competition had to classify 800,000 real web search queries into a maximum of five categories taken from a set of 67 categories designed by the competition organizers to broadly cover most topics found on the internet. To evaluate entries, the organizers picked a subset of 800 non-junk English queries and had them classified manually by three human labelers. They then ranked the 37 solutions based on overall precision and overall F1 value, as computed by Equations (6-10).

\[
\text{Precision} = \frac{\sum \text{queries correctly labeled as } c_i}{\sum \text{queries labeled as } c_i}
\]  

(6)

\[
\text{Recall} = \frac{\sum \text{queries correctly labeled as } c_i}{\sum \text{queries belonging to } c_i}
\]  

(7)

\[
F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\]  

(8)

\[
\text{Overall Precision} = \frac{1}{3} \sum_j \text{Precision against labeler } j
\]  

(9)

\[
\text{Overall F1} = \frac{1}{3} \sum_j \text{F1 against labeler } j
\]  

(10)

In order for our system to compare to the KDD CUP competition results, we need to use the same set of category labels and to implement the constraint of having a maximum of five categories per query. Our system uses a considerably larger and more detailed set of approximately 300,000 categories, and the set of KDD CUP test queries were classified into a total of 3,500 of these categories. However, since the KDD CUP category set is designed to broadly cover all possible topics, it is possible to map each of our categories to at least one KDD CUP category, and
to up to three categories to allow for ambiguous matches. We created this mapping manually for each of the 3,500 categories used by our system to classify the 800 KDD CUP test queries and then automatically limited the category set of each query to the five most frequently-occurring mapped categories. With this done, we computed the overall precision and F1 of both variants of our classification system following the KDD CUP guidelines. Our results are presented in Table I, along with the best KDD CUP system on Overall F1, Overall Precision, and the competition mean [1]. As we can see from these results, our system shows a 12% improvement in precision and an average 4% improvement in F1 over the competition mean. In the rankings, we made the top 10 for F1 value, and we got the third place for precision.

Table 1: KDD CUP Results

<table>
<thead>
<tr>
<th>System</th>
<th>Precision Ranking</th>
<th>F1 Ranking</th>
<th>Overall Precision</th>
<th>Overall 1 F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best F1</td>
<td>2</td>
<td>1</td>
<td>0.41</td>
<td>0.44</td>
</tr>
<tr>
<td>Best Precision</td>
<td>1</td>
<td>2</td>
<td>0.42</td>
<td>0.43</td>
</tr>
<tr>
<td>Mean</td>
<td>13</td>
<td>18</td>
<td>0.25</td>
<td>0.24</td>
</tr>
<tr>
<td>Ours</td>
<td>3</td>
<td>10</td>
<td>0.37</td>
<td>0.28</td>
</tr>
</tbody>
</table>

4.2 TREC 2007

The Text REtrieval Conference (TREC) is organized annually to support research in the field of text retrieval. From 1999 to 2007, it included a QA track, in which participants had to implement and demonstrate systems that could retrieve the answers to each of a set of questions from a large and varied text corpus. A complete technical report on the 2007 QA track and its results was compiled by Dang et al. [13]. We used the 445 questions from all 70 topics of that QA track as a test corpus for our system. While this gives us a good test corpus with which to experiment, to the best of our knowledge no one tried using the TREC data for query classification before. Consequently, we have no benchmark values with which to compare our results.

To evaluate our classification results, we began by manually creating a correct version of the query classification. In a manner reminiscent of [9], this was done by searching Wikipedia for the article that best matches the subject matter of the query and using the set of categories of that article as the correct classification. For some ambiguous queries that could relate to topics in several articles, we used the categories of all relevant articles. We then computed the precision and F1 value of our system compared to this correct classification in the same way as for the KDD CUP data, using Equations (6) to (8).

In this test, our system achieves both a precision and F1 of 0.22. This is a drop compared to the previous test, of 6% on F1 and of 15% in precision.

4.3 Discussion

As we pointed out, the results of the TREC test are notably lower than those of the KDD CUP test. However, while both tests followed the same methodology, we should note that the TREC test was a lot more difficult. Indeed, while the KDD CUP classification was limited to a set of 67 categories, the TREC test made use of Wikipedia’s entire set of 300,000 categories. Moreover, the correct classification used to evaluate the KDD CUP test was limited to a maximum of 5 categories and the average query had less than that, while in the TREC test we used the category set of Wikipedia articles as the correct results, and this set typically contains 7 to 10 categories per article. Our classifier, on the other hand, returns on average between 2 and 5 categories for a given query, on par with the KDD CUP expectations but well below the TREC ones. The considerably larger set of possible class labels and the discrepancy in the expected number of classification results both contributed to lowering our system’s performance in the TREC experiment.

Our results nonetheless show that our classifier can handle both web-style keyword searches and grammatically-correct complete human questions. This is an important fact to note, as these two types of queries are quite different. A typical web query is only a short list of keywords related to what the user is searching for. On the other hand, a human question is a complete sentence which includes words that are unrelated to the query but are needed for grammatical correctness or style, and these words can easily mislead a keyword-based search algorithm. These questions thus require some more sophisticated NLP techniques to discern relevant information. In our system, detecting the relevant information from longer statements is done by the combination of three steps. First, in the corpus preparation stage, only the significant keywords contained in the wikilinks of each article are kept. Next, the individual words of the user’s query are assigned a significance weight $R_w$. Finally, all query words have to be featured in either the title or the wikilink words of an article to be considered relevant. If no such article exists in Wikipedia, then some irrelevant words have been included in the query, and we iteratively shorten the list of query words until we’ve generated the correct list of keywords.

On the other hand, a problematic feature of web queries that is not present in proper English questions
comes from the fact that web users commonly make mistakes and typos. The KDD CUP data set, being composed of real web queries, does indeed include a number of typos, as well as non-English words, abbreviations, technical terms, and gibberish. In this regard, using Wikipedia gives our system a net advantage compared to using another knowledge base. While an English corpus such as an ontology, WordNet, or a regular encyclopaedia would only include the proper spelling of a word, Wikipedia includes its common typos and abbreviations as redirect links to the correct articles, and these links are included in our corpus preparation stage. Moreover, Wikipedia includes more technical terms and abbreviations than all but the most specialized technical resources [11]. As a result, our query classifier is able to recognize more words than a system based on a more limited resource, and can more precisely filter out gibberish. This advantage is not unique to our query classifier, but is shared by any NLP system built on Wikipedia, and was explicitly stated by [10] when discussing the success of their word disambiguation system. However, Wikipedia has not, to our knowledge, been used in this way for the purpose of query classification before. The only other query classification system based on Wikipedia seems to be that of Hu et al. [7], but they limited theirs to only three seed concepts, namely travel, personal names, and jobs, thus foregoing the advantage that comes from Wikipedia’s large and diverse topic coverage.

5 Conclusion and Future Works

In this paper, we presented a novel approach for query classification using encyclopaedic knowledge mined from the online encyclopaedia Wikipedia. Our method includes both a corpus preparation stage, which can be generalized to any online encyclopaedia, and a statistical classification algorithm that can be applied to any properly prepared corpus. By using Wikipedia in particular, our system gained the ability to classify queries into a set of 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 well as the ability to recognize and handle uncommon words such as technical terms and typos. The experimental results we presented showed mathematically that our system can handle both complete English questions and web-search-style keyword queries, and that it can classify queries nearly as well as the top classifiers found in practice.

For sure, more work remains to be done on our system to surpass the top literature benchmarks. In this regard, one last advantage of using Wikipedia is the wealth of untapped information we could add into our system to enrich it. For instance, while we only considered wikilinked words in our system, we could instead use all words in an article, possibly with a weighting scheme to differentiate the importance of wikilinked words compared to regular text. Moreover, instead of (or in addition to) considering wikilinked words more important, we could use one of the other metrics of word importance that have been proposed in Wikipedia-related research projects. One such metric was proposed by Ahn et al. [9], who observed that more relevant words in Wikipedia articles tend to occur earlier in the article’s text. Finally, we can note that there exists a hierarchical relationship between categories in Wikipedia, which our system thus far ignores. By using these connections, we could increase the value of different but closely-related categories in a query’s classification result set, and thus improve their rankings at the detriment of isolated unrelated categories that may have also been included in result set. Clearly, there is a massive potential for development and improvement of the system we proposed here.

6 References


