Search Query Categorization at Scale

Michal Laclavík
Magnetic Media Online
122 West 27th Street, 7th floor
New York, NY 10001
laclavik@magnetic.com

Marek Ciglan
Magnetic Media Online
122 West 27th Street, 7th floor
New York, NY 10001
marek@magnetic.com

Sam Steingold
Magnetic Media Online
122 West 27th Street, 7th floor
New York, NY 10001
sam.steingold@magnetic.com

Martin Šeleng
Institute of Informatics
Slovak Academy of Sciences
Dúbravská cesta 9, Bratislava
martin.seleng@savba.sk

Alex Dorman
Magnetic Media Online
122 West 27th Street, 7th floor
New York, NY 10001
alex@magnetic.com

Štefan Dlugolinský
Institute of Informatics
Slovak Academy of Sciences
Dúbravská cesta 9, Bratislava
stefan.dlugolinsky@savba.sk

ABSTRACT

State of the art query categorization methods usually exploit web search services to retrieve the best matching web documents and map them to a given taxonomy of categories. This is effective but impractical when one does not own a web corpus and has to use a 3rd party web search engine API. The problem lies in performance and in financial costs. In this paper, we present a novel, fast and scalable approach to categorization of search queries based on a limited intermediate corpus: we use Wikipedia as the knowledge base. The presented solution relies on two steps: first a query is mapped to the relevant Wikipedia pages; second, the retrieved documents are categorized into a given taxonomy. We approach the first challenge as an entity search problem and present a new document categorization approach for the second step. On a standard data set, our approach achieves results comparable to the state-of-the-art approaches while maintaining high performance and scalability.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]; H.3.1 [**Content Analysis and Indexing**]; H.3.3 [**Information Search and Retrieval**]

Keywords

query categorization; search; indexing; Wikipedia

1. INTRODUCTION

Query categorization (QC), or query classification, is a task of identifying user search intent based on a submitted query and mapping it to predefined categories. Query categorization can be used for several applications; the most notable one is, probably, online advertising. In the online advertising domain, the QC can help capture user interests and improve user modeling, which in turn can lead to an increase in precision of user targeting with ads relevant to their interests and needs. The motivation for this work came from the domain of search retargeting, a form of targeted advertising where the audiences are modeled based on the search queries users conduct on websites they visit. By modeling user interests, the query retargeting has the ability to find new customers, who never visited a marketer's website before. Search retargeting focuses on displaying advertisements to users who conducted searches for specific keywords or categories in the past. For this domain, QC is the essential technique for user modeling and better user targeting.

The task of query categorization is quite intricate. One has to map a short string, representing a query, to another short string, representing a category, while the two strings usually have only marginal lexical similarity or none at all. For this reason, the common approach is to extend both queries and the categories by gathering additional information, richer in textual content. The richer textual representations are then compared. The most popular method to gather additional data for queries is to retrieve results a web search engine yields for the given query. The documents and, possibly, categories retrieved by the web search are then compared with representations of target categories. When one does not own a large web crawler covering a significant part of the web (an asset only a few organizations have), the only option is to use an API of a web search product. This approach is impractical due to the financial costs and for performance reasons.

Meanwhile, companies focusing on search retargeting gather large numbers of user-generated queries that need to be categorized, on the order of thousands of queries per second. This challenge really needs a scalable and fast approach independent of a web search engine API. The research question we try to address is whether we can use a relatively small (compared to a web corpus), off-line corpus as a means of information extension and still retain results competitive with the methods relying on the Web as a corpus. Wikipedia is suitable as the intermediate corpus because it is relatively small (under 40 GB of text) and has a very broad coverage. The proposed solution relies on two steps:

1. Extend search query with information from Wikipedia.
2. Map extended query data to pre-defined categories.

WWW 2015 Companion, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3473-0/15/05.
http://dx.doi.org/10.1145/2740908.2741995.

We approach the first task, the query extension, as an entity search problem. A search query is mapped to Wiki pages and then results are enriched with data from semantic knowledge bases DBPedia and Freebase. These pages are then classified based on the given taxonomy.

Main contributions of the paper are following:

- We propose a query categorization approach that uses limited data resources and is scalable, while achieving quality of results competitive with state-of-the-art approaches that use the entire web as a corpus for information extension.
- We approach the challenge of the query information extension as an entity modeling and search problem. This part is not in the scope of this paper and is covered in the write-up [6] from the ERD challenge focusing on entity recognition in search queries, where we participated and achieved high scores.
- We propose a text classification method, which defines categories by representative n-grams with scores. N-grams are derived by reusing human knowledge encoded in Wikipedia articles using the Wikipedia link graph. Subsequently we are using a simple N-gram matching technique to categorize text.

The paper is structured as follows. Section 2 summarizes scholarly works related to the search query categorization task. We outline the proposed solution in Section 3. Details from query and corpus modeling are covered in [6] and we give a short overview in Section 4. The novel document categorization and its impact on query categorization is the main part of paper and is described in Section 5, where we study the effects of different options of the categorization process on the end-to-end results quality. We evaluate the scalability of the proposed solution in Section 6.

2. RELATED WORK

Most of related scholarly works were triggered by the ACM KDDCUP 2005 competition[1], which was focused on query categorization. The goal of the challenge was to categorize provided queries into 67 categories, organized as a two-level taxonomy[2]. A subset of 800 queries was manually categorized by three human labelers, providing a test set for the task evaluation. It is worth mentioning the findings reported in [10], where each labeler was evaluated against the other two labelers, which showed that average value was 50.9% F_1 and 52.6% Precision. This indicates that QC is a hard task for humans as well, thus results achieved with automated systems can hardly be better.

The winning solution of the KDDCUP 2005, by Shen et al [9], achieved 41.4% precision and 44.4% F_1 score. The same authors later improved the method in [10] achieving 43% precision and 46.1% F_1 score in the best run. This is currently the best known result on the 2005 KDDCUP data set, using search engine, intermediate taxonomies and SVM-based classification. Similar results, with 46% of F_1, are reported in [2] using queries logs from a web search engine and documents retrieved by the web search. All of these best-performing solutions use the entire Web as a corpus via a search engine or search engine logs. Such an approach can work well for web search companies, who have a web search engine at hand. Unfortunately, it is slow and costly when using a web search engine over a paid API. Another recent work [11] claims significant improvements over existing approaches, using an interesting approach of learning from Dmoz categories. On the KDD CUP dataset it achieves high Precision of 59.5% but with a very low F_1 of 33.1%. Again the entire web was used as corpus for query categorization. All these approaches focus on mapping retrieved documents to categories treating it as a text categorization problem via machine learning, intermediate taxonomies or document classification, but leave query modeling, which is a large part of the query categorization task, up to web search engine.

Several works modeled queries using Wikipedia. In [4], Wikipedia data (text, link graph) are used to model user's query intent. However, the query intent is evaluated only on 3 categories, achieving quite high scores running in binary mode, where the algorithm decides if a query falls in a category or not. It is not clear how it would behave on a larger taxonomy. Another work [5] uses Wikipedia as a corpus for the query categorization task, utilizing the information retrieval approach based on a vector space model, but the method achieved only 32.2% F_1 on KDDCUP data.

3. QUERY CATEGORIZATION USING INTERMEDIATE CORPUS

Our hypothesis was that a large number of the queries can be answered to some extent by Wikipedia to get an understanding of query categories. There are, of course, queries not covered by the Wikipedia content, e.g. some company or product names. We have considered other data sources such as Freebase and DBPedia. We use semantic information from both knowledge bases to enrich Wikipedia concepts. Namely, we exploit Freebase and DBPedia types (ontological hierarchy of categories) associated with concepts.

Our solution relies on two high level steps:

1. We extend the query with the information from Wikipedia corpus, by mapping the query to relevant Wikipedia concepts.
2. We map Wikipedia concepts associated with the query to items in the taxonomy.

For the first step, the extension of query information, we took inspiration from scholarly works on entity search and we treat the challenge as an entity search problem and we successfully turned queries to Wikipedia entities by participating in the *2014 Entity Recognition and Disambiguation Challenge*[3]. We have participated in the *Short Track* of the challenge, which focused on recognizing mentions of entities in search queries, disambiguating them, and mapping them to the entities in a given knowledge base. In the ERD, our system was evaluated as the 4^{th} best, with an F1 score of 65.57%[4]. Recognizing Wikipage concepts in queries is a first step of our QC solution.

The second step of the approach is concerned with mapping Wikipedia concepts associated with a query to target categories. Each category in the input data set is repre-

[1] `http://www.kdd.org/kdd-cup-2005-internet-user-search-query-categorization`

[2] In practice, our taxonomy is based on several publicly available taxonomies (e.g., IAB), adapted to the business needs of search retargeting (i.e., who we want to advertise to) as well as the actual query distribution (i.e., we tried to avoid empty and overly abundant categories), which resulted in just over 400 categories organized in 3 levels. The experiments in this paper, however, use the KDD taxonomy.

[3] `http://web-ngram.research.microsoft.com/erd2014/`

[4] `http://tinyurl.com/ShortTrackERD14`

sented by a short string with very limited information content. Thus, we extend category strings with information from the Wikipedia. For each category, we select several representative Wikipedia pages related to the given category. Having enriched query and categories with Wikipedia documents, we basically need to solve a document categorization problem. We have evaluated multiple techniques, including tf-idf similarity, LSI [8] and a gazetteer-based classification. We discuss the second step in detail in Section 5.

4. MODELING QUERY WITH CORPUS DOCUMENTS

In this section, we briefly describe the process of enriching a query with the data from the intermediate corpus. As already mentioned, we have used Wikipedia instead of the entire Web and we have created an entity search solution based on Wikipedia, Freebase and DBPedia data which is described in [6]. To summarize this approach, we built a search index using the Lucene library, where each document is a Wikipedia article broken into a variety of fields such as title, alternative names, links, abstract, or categories from Wikipedia, Freebase and DBPedia. There are 17 fields used altogether. Each query is searched in the index, and subsequent post-processing is done to eliminate and disambiguate candidates. Post-processing uses Wikipedia link graph and alternative names mapping. For Example, the query "total recall schwarzenegger" is modeled as several Wikipage documents including two Total Recall movies, and several people with the Schwarzenegger surname, but the final output of the algorithm is Arnold Schwarzenegger[5] and Total Recall[6] Wikipages, which are then categorized.

In addition to representing queries by Wikipedia entities, which is achieved by our entity search approach [6], we also detect n-grams representing categories directly in queries. The majority of queries misinterpreted by our initial approach are *type queries* (according to the terminology used in [7]), as they contain terms strongly associated with a category (e.g. "Louisiana state *jobs*" or "funny *videos*"). The "type queries" require a different approach, as the standard "entity search" approaches often fail to get the correct results. The type queries state their "type" directly – they contain terms characteristic of a category within the query itself. Those queries are quite easily categorized by using simple matching of keywords characteristic of distinct categories, when good keywords (n-grams) are available. We describe how we generate good n-grams to represent categories in section 5.2.

The impact of detecting n-grams directly in query is shown in Figure 1. To give a concrete example, consider the "total recall *movie*" query. Here we would simply detect the query category by noticing the "movie" n-gram, which is strongly related to the "Entertainment/Movies" category.

4.1 Algorithm Specification

In this section, we describe the algorithm for combining search result scores of Wikipedia entities representing queries. When "Arnold Schwarzenegger" and "Total Recall" are found for the "total recall schwarzenegger" query, we categorize them and get categories with scores for each Wikipage document, which need to be combined to return the query category. (We combine n-grams in the categorization process the same way).

Given a query q, we search for it in the corpus, producing matching documents D and similarity scores $s > 0$. Each document D is assigned to categories c with probabilities p. We assign categories to the query as follows.

We start with filtering the set of matching documents for the each query by discarding all the matches which have a similarity score below the square root of the best similarity score to reduce the number of returned documents.

Next, we treat the similarly scores $s > 0$ as if they were odds and interpret them to mean $\mathbb{P}(R(q, D)) = s/(1 + s)$ where $\mathbb{P}$ is probability and $R(q, D)$ means that query q is related to the document D.

We also assume that the matching of queries and documents is probabilistically independent from the assignment of documents to categories, therefore

$$\mathbb{P}(A(q, c)) = 1 - \prod_{D}(1 - \mathbb{P}(R(q, D))\,\mathbb{P}(A(D, c))) \quad (1)$$

where $\mathbb{P}(A(x, c))$ is the probability that entity (query or document) x should be assigned to category c.

Note that adding an extra document, which is either unrelated to q (i.e., $\mathbb{P}(R(q, D)) = 0$), or should not be assigned to c (i.e., $\mathbb{P}(A(D, c)) = 0$), to the product above does *not* change the value of the product, so the result is well-defined.

5. DOCUMENT CATEGORIZATION

Our approach relies on modeling a user query with Wikipedia concepts and subsequently classifying text documents associated with given Wikipedia concepts to target categories. For the document classification (DC), we have evaluated several different approaches:

1. classical DC approaches (tf, tf-idf, LSI),
2. DC based on n-grams characteristic for categories,
3. taxonomy bridging (mapping DBPedia and Freebase entity types to target categories.)

5.1 Exploiting vector-based similarities

Document categorization is a well studied field of computer science. We have evaluated several standard methods used in the literature for the document classification task in our problem domain. We have built a vector representation of documents and we have used the cosine similarity to determine the document-to-document similarity. We experimented with a) term frequency (tf) vectors, b) tf-idf vectors, c) topical vectors generated by latent semantic indexing (LSI) [8].

We need to compute a similarity of a given text to a category from a given taxonomy. We have to represent categories by other, richer means than category name. In this work we represent categories by Wikipedia articles. We manually assigned representative Wikipedia articles to distinct categories; we will refer to articles assigned to a category as "category seed documents". This labor-intensive process can be assisted by exploiting the Wikipedia index search. It is a one-time job (for each taxonomy) and the time required for manual assignment is not prohibitive (e.g. for a taxonomy size similar to the one used in KDD CUP 2005 data set – 67 categories[7] – the job can be done in under two hours). For example, for the *Computers/Mobile Computing* category we

[5] http://en.wikipedia.org/wiki/Arnold_Schwarzenegger
[6] http://en.wikipedia.org/wiki/Total_Recall
[7] http://ikt.ui.sav.sk/research/QC/KDD_wikipedia.txt

have assigned the following seed Wikipedia articles: *Mobile computing*; *Smartphone*; *Mobile phone*; *Camera phone*.

The text of "Wikipedia seed articles" was used as a text representing categories in our experiments with standard categorization methods (tf, tf-idf, LSI). To categorize an input text into a given taxonomy, we first transform the text to its vector representation. Then, we compute the similarity of the text vector to vectors of representative category documents. Categories related to the top-scoring vectors are considered to be the categories related to the input textual content. We have used tf, tf-idf and LSI vectors to represent texts as vectors. We did not directly evaluate the Wikipedia page categorization; rather we measured the effect of different categorization approaches on the overall QC process. With all three evaluated methods, we were achieving results lower than 30% for the F_1 score.

5.2 Representing Categories by Keywords

Methods described in the previous section operate on the level of terms, single words. To capture richer information from the input text, we were looking for an option that would exploit term n-grams. We use the term n-gram to denote an ordered sequence of words commonly used together to denote or describe a concept. The underlying idea is that n-grams represent categories much better than single words contained in an n-gram; e.g. presence of 2-grams "real estate" or "human rights" represents particular categories very well, while being very generic when used as single keywords. Additionally, n-grams can be used to detect the category directly from the query. The principal challenge is how to obtain n-grams that are representative for a given category. In our previous work [1], we extracted n-grams from Wikipedia using the indexing and search approach, and categorized documents into DBPedia categories. Here, we propose a new way of extracting n-grams based on Wikipedia link similarity and utilizing them for document categorization. As discussed in the previous subsection, we have manually assigned "seed Wikipedia pages" to all categories, which provided us with a rich textual representation of categories. Our initial approach was to use existing keyword extraction tools to mine the representative keywords and n-grams directly from the text. However, we were not able to achieve results competitive with methods discussed in the previous subsection.

Instead of extracting representative keywords and n-grams from the text, we have turned to the link-graph of Wikipedia. The link graph represents the structure formed by the hyperlinks between Wikipedia articles. A link usually denotes a relationship between the two linked concepts. In fact, it encodes human knowledge on the relation of the given concept with the rest of the content in the Wikipedia. We have generated the set of concepts related to the seed concepts of a category as follows:

- For a seed article A, we construct a set of concepts $s(A)$ that A links to.
- Based on the DBPedia and Freebase categories, we remove from $s(A)$, all the concepts of the types: *Location*, *Person*, *Organization* and *Work* (this resulted in fewer n-grams and had none or minor impact on results; additionally, *Location*-based similarity was misleading).
- For all remaining members of $s(A)$, we compute the cosine similarity with the concept A, where we use link adjacency as vectors for the computation. The computed value is used as a concept similarity score to the target category.
- We construct a list of pairs $<$ name, score $>$, where name (n-gram keyword) is the name of the Wikipedia concepts and score is value of its cosine similarity with a seed category concept.
- We extend the list by the alternative names (defined by titles of Wikipedia redirect pages pointing to the concept) of each concept.

An example of a small portion of a list produced for category *Computers/Mobile Computing* is provided below. The score 1.0 indicates that the concept has been assigned as a seed concept (in this case 4 seed articles have been assigned). The rest of the concepts are derived from the neighborhoods of the seed concepts in the Wikipedia link graph. The score is the value of the cosine similarity of the adjacency vectors. (The example provided below does not contain list extension by alternative names of the concepts.)

```
Computers/Mobile Computing
        Mobile computing        1.0
        Mobile phone    1.0
        Smartphone      1.0
        Camera phone    1.0
        Mobile operating system 0.3413
        Feature phone   0.2679
        Android (operating system)      0.2098
        ...
```

In this way, we have generated a list of keywords and n-grams related to different categories. Such an approach allows us to unlock knowledge encoded in Wikipedia to represent categories by meaningful n-grams with probability scores. Based on the produced data set, we have developed a simple classification algorithm, described in the next section.

5.3 Text Classification Procedure

Our classification method consists of applying the algorithm described in section 4.1 to n-grams instead of queries, followed by combining the n-gram categories into the query categories, and then filtering out low probability categories.

5.3.1 Combining Categories

We assign a query to a category if we assign any of its n-grams to the category. This means that, as in (1),

$$\mathbb{P}(A(q,c)) = 1 - \prod_g (1 - \mathbb{P}(A(g,c))) \qquad (2)$$

Observe that if a category cannot be assigned to any n-gram (i.e., $\forall g\, \mathbb{P}(A(g,c)) = 0$), then $\mathbb{P}(A(q,c)) = 0$, and if we are certain of an assignment of an n-gram to a category (i.e., $\exists g\, \mathbb{P}(A(g,c)) = 1$), then $\mathbb{P}(A(q,c)) = 1$.

Note that the same formula (2) is used to combine categories from Freebase and DBPedia with the n-gram categories (see section 5.4).

Implementation Remarks: The approach for n-gram matching is a simple gazetteer-based solution used in information extraction. We have used an implementation based on an in-memory char-tree, where all n-grams were loaded into memory creating an efficient character-tree structure [3]. This allowed us to have linear performance in n-gram detection in text, where we read text representing the Wikipedia article only once, firing detected n-grams and categories represented by a score.

Table 1: Evaluation of Solution on Wikipedia index with various approach for Categorization

	TF Cos-Sim	Tf-idf Cos-Sim	LSI	DB, FB	n-gs text	n-gs sent.	n-gs abst.	n-gs abst. & type	DB FB
P	15.1	27.0	17.7	33.7	31.4	34.2	33.7	45.9	46.7
R	24.3	31.6	29.4	30.4	41.5	38.7	40.7	35.1	41.8
F_1	18.4	28.8	21.8	31.6	35.5	36.0	36.6	39.5	43.8

5.4 Mapping different categorization schemes

In addition to n-gram keywords, we have also provided a manual mapping of DBPedia and Freebase categories to KDD categories. For example, we have mapped DBPedia *Celebrity* or *MusicalArtist* category into *Entertainment/Celebrities* category. We have experimented with document categorization also using these mappings alone or in combination with n-gram keyword categorization. Results are combined by applying the same formula (2) as when combining scores of detected n-grams or categories of search results. Experiments are described in the next section.

5.5 Evaluation of Categorization

In this section, we evaluate various categorization algorithms and their settings in the context of the whole QC process. We discuss how different options influenced the quality of results. We summarize results of different tested document categorization approaches in Table 1 and visualize them in Figure 1. In Table 1, the first three columns show cosine similarity of 1) term frequency (TF) vectors, 2) Tf-Idf vectors, and 3) LSI vectors, when applied in our query categorization process (methods described in Section 5.1). Categories were represented as text of relevant Wikipedia articles. With TF vectors, results were quite low; with Tf-idf we have reached almost 29% for the F_1 score. Surprisingly, the usage of LSI vectors yielded quite low scores as well. The fourth column of Table 1 shows the results of categorizations using the manual mapping of DBPedia and Freebase categories to KDD categories (the method is described in the previous section 5.4). F_1 was almost 31%, which is better than evaluated text categorization techniques. The highest score was achieved when using the n-gram matching technique (with automatically generated keywords and n-grams, process described in Section 5.2). We believe this is mainly because the n-grams were identified based on the human knowledge encoded in the Wikipedia topology.

We have tested suitability of various representations of Wikipedia pages for the categorization task. In Table 1 and Figure 1, we can see evaluation of various n-gram matching settings, where we have tested categorization on the text of whole Wikipedia articles, the first sentence of an article or the article abstract. The best results were achieved with article abstract. Using the whole Wikipedia page text gave us sometimes misleading results. The inclusion of larger content often brings more noise and marginally related topics are often detected. For example, a document about a person might be categorized within the *Football* category, while the person's main occupation has nothing to do with football. This can happen just because the Wikipedia article mentions that the person was playing football for the college where he studied. We have tried to use just the first sentence of Wikipedia articles. This approach did not work especially well; the main reason being that often the first sentence contained very little information. This is why we have ended up using Wikipedia abstracts, achieving better results compared to a full article text or just the first sentence. The "type query" approach of the query understanding workflow was turned on in the `n-gs abst & type` column. The important fact is that the results of the process for generating characteristic n-grams for categories can also be exploited for "type query" of the query understanding workflow. The `n-gs abst & type` column of Table 1 shows the score where we have used generated category n-grams for document classification and also for the type query mapping (detecting category-related n-grams directly in query). This led to further improvements in the F_1 score. With n-gram categorization techniques we have reached 39.49% of F_1 on the KDD dataset. Since we have also done manual mapping of Freebase and DBPedia categories, we have tested how they can be used in improving categorization results. We have combined the n-gram matching technique with the taxonomy mapping technique. The results are summarized in last columns of Figure 1 and Table 1. We have used DBPedia and Freebase (DB/FB) categories only for a limited number of categories that could not be detected well by the automatically generated n-grams. E.g., n-gram based classification did poorly for identification of the *Celebrity* category; whereas DBPedia/Freebase categorization was very helpful for this particular category. Similarly, the usage of DBPedia/Freebase categorization yielded very good results for *Shopping*, *Travel* and *Local & Regional* categories.

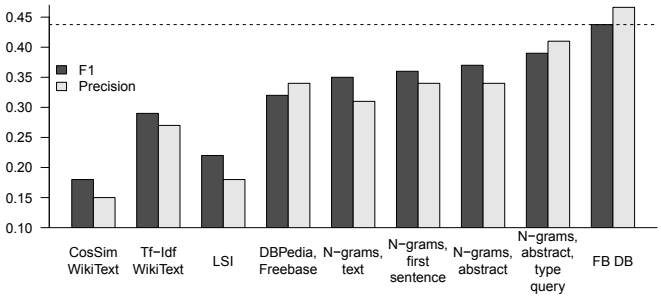

Figure 1: Various categorization approaches

We have used a mapping limited to a small subset of categories for which the n-gram matching solution performed poorly. With the limited category mapping, we have almost achieved 44% F_1 score. This result is comparable with state-of-the-art approaches that use the whole web as the intermediate corpus. We also provide the data from our experiments and model used on the paper support page[8], to facilitate reproducibility of the experiments.

6. SCALABILITY

The goal of the presented research effort was to provide a query categorization approach that would deliver decent results and would be scalable to process thousands of queries per second.

We tested the speed and scalability of our solution. Evaluation was done on a single machine with the following configuration: Intel(R) Xeon(R) CPU E5-2620 @ 2.00GHz; 2x6 cores processors; 32GB RAM. We have used the popular

[8] `http://ikt.ui.sav.sk/research/QC/`

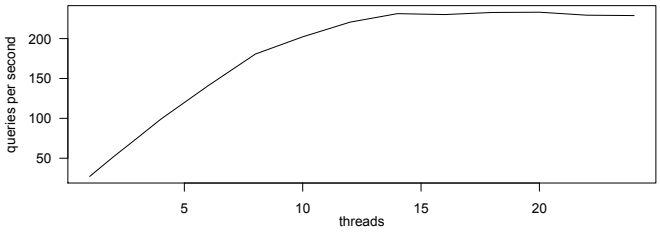

Figure 2: QC scalability with multiple threads

information retrieval library Apache Lucene to create and interact with the index of the intermediate corpus. The index was loaded in memory to avoid costly disk operations. The developer version of our Wikipedia index has about 14 GB, when all information for on-line categorization is stored in the index (by online categorization we mean categorizing Wikipedia pages on the fly, when retrieved from the index). A production index with pre-computed categories was smaller in size, taking about 8.5 GB; it can be loaded in memory on a standard machine.

When running QC in a single thread, one query is categorized in about 37 milliseconds. With a number of threads, it scales almost linearly with number of available cores. We have evaluated the solution with up to 24 threads, since we have 12 physical cores with hyper-threading. In Figure 2 we summarize the results. It shows that Lucene scales well and can categorize about 230 queries per second on a machine with 2x6 cores when running on a number of threads a bit higher than the number of cores. For an experiment with a query log containing 3,000 unique queries and with 14 threads, one query is categorized in 4.3 milliseconds. We have also evaluated the solution on 1 million queries taken from AOL query logs[9], where some queries are repeated. We have run experiments on the same machine with 14 threads. One million queries were categorized in 57 minutes, so one query was categorized in 3.41 milliseconds. The presented approach could handle about 293 queries per second.

When this approach was deployed in production using Apache Solr, it was able to categorize in average 400 queries per second on one server. When combined with a Varnish caching server, it stabilized at around 2,500 queries because of repeating queries in real workloads. Additional boxes can be added to scale it horizontally.

7. CONCLUSION AND PERSPECTIVE

In this paper, we describe a fast and scalable method for search query categorization. While the best performing solutions use the entire web as a corpus, we proposed a solution based on simple information retrieval methods using Wikipedia as a corpus for query categorization, and categories represented using scored n-grams. The F_1 score of our method is 5.1% lower than the best known approach (43.76% ours; 46.1% the best) but with an 8.4% increase in precision (46.63% - ours; 43% - the best). While web search-based approaches can be applied effectively by several major companies, our approach is fast and scalable with limited data resources

The main contributions lie in: 1) scalability 2) applying entity search on QC task, where query is modeled by Wikipedia instead of entire web, as well as 3) novel n-gram categorization method, where n-grams are detected based on human knowledge encoded in Wikipedia.

In real settings, we categorize 3 types of searches: natural searches, navigational searches, and page keywords. In Online Advertising, it is important to serve relevant ads and not waste resources or upset users by irrelevant ads. This is why the algorithm is tuned to higher Precision, above 70%, in production, even when reaching lower Recall. Wikipedia does not cover all topics well, but it still contains a huge amount of human knowledge, including well known products and brands. Even when a search contains entities not covered in Wikipedia, the query often contains keywords related to generic Wikipedia concepts or n-grams directly related to category, which can still be categorized well. For under-performing domains the knowledge base can be extended with additional sources. An example of well-covered domain in Wikipedia is Automotive, where we are able to categorize queries with both Precision and Recall over 90%.

[9] `http://jeffhuang.com/search_query_logs.html`

8. ACKNOWLEDGMENTS

This work is supported by Magnetic and by projects: VEGA 2/0185/13, CLAN APVV-0809-11, FP7-284984. We would like to thank Tom Comerford for editing the paper.

9. REFERENCES

[1] M. Ciglan, M. Laclavik, and A. Dorman. Reusing knowledge hidden in wikipedia for scalable text categorization. In *Proceedings of WSDM Workshops: WSCBD*, WSDM'14 Workshops, 2014.

[2] E. Diemert and G. Vandelle. Unsupervised query categorization using automatically-built concept graphs. WWW '09, pages 461–470, 2009.

[3] S. Dlugolinsky, G. Nguyen, M. Laclavik, and M. Seleng. Character gazetteer for named entity recognition with linear matching complexity. In *Proceedings of WICT*, WICT'13, pages 364–368, 2013.

[4] J. Hu, G. Wang, F. Lochovsky, J.-t. Sun, and Z. Chen. Understanding user's query intent with wikipedia. WWW '09, pages 471–480, 2009.

[5] M. Kouylekov, L. Dini, A. Bosca, and M. Trevisan. Wikipedia-based unsupervised query classification. In *IIR*, pages 116–119, 2013.

[6] M. Laclavik, M. Ciglan, A. Dorman, S. Dlugolinsky, S. Steingold, and M. Šeleng. A search based approach to entity recognition: Magnetic and iisas team at erd challenge. In *Proceedings of the ERD '14*, pages 63–68, New York, NY, USA, 2014. ACM.

[7] J. Pound, P. Mika, and H. Zaragoza. Ad-hoc object retrieval in the web of data. WWW '10, pages 771–780, 2010.

[8] R. Rehurek. Subspace tracking for latent semantic analysis. ECIR'11, pages 289–300, 2011.

[9] D. Shen, R. Pan, J.-T. Sun, J. J. Pan, K. Wu, J. Yin, and Q. Yang. Q2c@ust: Our winning solution to query classification in kddcup 2005. *SIGKDD Explor. Newsl.*, 7(2):100–110, Dec. 2005.

[10] D. Shen, J.-T. Sun, Q. Yang, and Z. Chen. Building bridges for web query classification. In *Proceedings of SIGIR Conference*, SIGIR '06, pages 131–138, 2006.

[11] P. V. Ullegaddi and V. Varma. Learning to rank categories for web queries. CIKM '11, pages 2065–2068, 2011.