LAUREA MAGISTRALE IN INFORMATICA - COMPUTER SCIENCE

UNIVERSITÀ CA' FOSCARI VENEZIA

AN AUTOMATIC TOOL FOR LABELING WEB SEARCH MISSIONS USING WIKIPEDIA CATEGORIES

Relatore:	Chiar.mo Prof. Salvatore Orlando
Correlatore:	Dr. Gabriele Tolomei
Studente:	Marco Gezzele
Matricola:	810845

ANNO ACCADEMICO 2012/2013

Ai miei genitori

e ai veri amici

Contents

List of Figures

List of Tables

Abstract

Mission detection aims to identify the sets of queries that users submit to a Web search engine in order to satisfy common information needs. Moreover, it is generally easier to figure out the actual search topic a user is interested in by leveraging signals coming from several queries (i.e., mission) instead of looking at each query separately. In this work, we present a system that automatically labels the search missions previously discovered from a search engine log using a set of predefined semantic categories, as provided by Wikipedia. Those are well-known categories, which have been already proposed in previous work since they cover almost every topic underneath search missions. Our solution consists of the following steps. First, we extract the set of Wikipedia articles (i.e., entities) from each single search mission using a state-of-the-art entity linking technique. This is achieved by representing a search mission as a virtual text document; such document is made of the queries composing the mission as well as the text included in the web page, which the user possibly clicked on in response to each query. Second, we retrieve the set of candidate Wikipedia categories that correspond to the set of entities extracted during the previous step. Finally, we rank the predefined set of target categories with respect to the candidates above using an unsupervised approach, and we therefore assign the highest ranked target category to each search mission. In our experiments, we use a dataset of 8,800 queries sampled from a real-world search engine log. Furthermore, such queries were already *manually grouped into individual search missions, which in turn we use as input to our system. To evaluate the quality of our proposed solution, we conduct a user study where users are asked to manually evaluate the correctness of the labels assigned to the missions. This way it is possible to judge the goodness of our approach.*

Chapter 1

Introduction

Nowadays users are exploiting web search engines (WSE) to complete any kind of task and to satisfy any kind of information need. Whether a user is looking for knowledge, to plan a trip, or to buy something, a web search engine can help him achieve his objective. Because of this, WSE query logs, became more and more important to help understand how users search the web, what behaviour patterns they follow, and how it is possible to exploit these information.

Users though are not usually able to satisfy their information need with a single query, and often perform further queries in order to either refine, or to analyse different aspects of the same objective they want to achieve.

A search mission is then a chain of queries, either interrupted or not, submitted by a single user, with the aim of reaching the accomplishment of the same general task. As a simple example, the queries, *"car rental new york" "Hotel new York"* and *"flight for ny"* all belong to the same mission, that has the task of finding the necessary informations for a travel to New York.

The Mission detection task then is the job of identifying a missions by analysing web search engine query logs and grouping together queries that a single user submits to a WSE (web search engine) in order to satisfy a single information need.

Once the queries belonging to a Mission have been identified, we need to understand what the information need of the mission is about. Mission labelling aims at categorizing search missions by assigning them a label from a predefined and limited set. By using the preceding example again, we can see that the 3 queries *"car rental new york" "Hotel new York"* and *"flights for ny"*, belong to the same mission, but we still need to understand what the mission is about. By using a predefined limited set of labels, we can assign to the mission the label "Travel". Once the appropriate label is assigned it might be possible for a WSE designed for the purpose, to exploit it to further refine our research, and to direct the queries to the most appropriate vertical search engine, in this case one addressing travels.

Moreover, in mission labelling it is generally easier, compared to query labelling (the process of categorizing single WSE queries), to figure out the actual search topic a user is interested in. In fact while trying to label search missions, we can leverage information extracted from several queries instead of having to look at each query separately. Therefore all the queries belonging to a single mission help us contextualize what the mission is actually about. As an example, the preceding query "car rental new york", is clearly aimed at finding a rental car in New York, but it gives no information on whether it is because his car broke down (and therefore we might refine the search by looking for cheap mechanics) or if he needs the car for a trip. By exploiting the combined information of the three queries it is possible to understand that his searches were aimed at looking for travel informations, and therefore we can refine the searches accordingly.

Mission labels, if correctly assigned, can then be used to automatically refine future searches that fall within the same search mission of other past queries and, as a consequence, it might be useful tu subdivide the queries to different vertical web search engines, each specialized in a particular field.

Recent works though focused mainly on query labelling instead of mission labelling, but the obtained results did not reach sufficient precision levels to allow us to use those categories for query refinement. This is caused by the nature of the web search query, which are very short, and therefore it is very hard for an automatic categorization tool to properly understand the real information need behind it. This difficulty in understanding the information need is caused mainly by words ambiguity and by the fact that, without any additional information on the actual meaning of the query, disambiguation is extremely hard.

Exploiting different data sources, such as in web search missions, is therefore a useful and sometimes necessary step in order to understand the real information need.

Mission labelling then gives the chance to exploit a set of queries with the aim to correctly understand what the user is really looking for.

Mission labelling though is not at all perfect and still suffers from many issues:

- Queries are short, often misspelled or plain wrong (users often make mistakes, even repeatedly).
- Even if the queries are not wrong, they can often present ambiguities that might be hard to solve even with the help of additional data (what is clear and unambiguous in a user mind, might be impossible to understand for someone else).
- The users might not know precisely what they are looking for, making the search and navigation behaviour erratic, hence including possible wrong information in the navigation log.

- A category database that can properly represent all of the possible subjects that might be the aim of a search mission is needed in order to properly label the missions with meaningful categories, while avoiding being too general or too specific.

In this work, we present a system that automatically labels web search missions occurring in a web query log composed of both queries and navigation data. The dataset consists of of 8,800 queries manually grouped into individual search missions. We then use a set of 99 predefined semantic categories, extracted from the Wikipedia Category Graph in order to label the missions. Those are well-known categories, which have been already proposed in previous works. This 99 category set is used since it covers almost every topic underneath search missions, while trying to avoid being too general.

The choice of implementing a tool for search mission labelling was due to two main reasons.

- Mission labelling is a subject of which has not been talked about extensively in scientific researches.
- Mission labelling, compared to query labelling, gives us the chance of exploiting a higher quantity of data and information. It also gives us the chance of properly making our labelling choices based also on the context, which is nearly impossible when handling single, and probably short, queries.

Our solution for mission labelling consists of the following general steps, that will be further explained in the thesis.

As a first step, we build the text base from which we will link the Wikipedia articles (i.e., entities)in order to give semantics to the missions by associating with their search topic. This is done by not only using the queries associated with a single mission, but also leveraging the other information we might have, which is the homepage of the website visited by the user during the mission (we don't have any information regarding the specific web-page though). We therefore proceed by extracting text and information from the web-pages, and by exploiting it while searching for entities to extract from the mission data. To our knowledge, this type of approach to mission labelling has never been attempted.

Entity identification and linking in these mission text is implemented by using a state-of-the-art entity linking technique developed by Ceccarelli et al. [15]. This step returns a set of disambiguated entities from the Wikipedia database. Those entities though, might still be wrong, because of a possibly wrong query or of an extracted mission text which contains other entities that do not represent the mission properly. Other steps are then needed. In order to choose only the meaningful entities, we devised an evaluation algorithm aimed to assign a score to each entity based on a measure that defines the importance of a spot as a mention of an entity in the text. This first measure is used as a baseline and is then refined through a set of operations that aim to emphasize not only the entities with the highest link probability, but also the entities found in the web-pages that are more likely to be talking about the same subject as those in the queries.

In the following step we proceed to extract from these entities the corresponding Wikipedia categories to which they belong. Again we try to discover if different entities identified in a mission belong to the same categories, in order to exploit possible underlying subject relationships, and we then assign new scores to these categories based on both the entity they derive from, the co-occurrence of identical entities in other parts of the mission text, and from the presence of other identical categories. Once we have a category set with the relative scoring, from which to start the exploration of the Wikipedia Category Graph, we search for the closest "*Goal Category*" for each category in this set. in the Wikipedia Category Graph, though we exclude from the starting set of categories, those that do not meet a certain score threshold, in order to eliminate those categories.

Finally, we obtain a set of goal categories, ordered by the number of steps it took to get from the starting category to the goal one. Because of the high density of the Wikipedia category graph, even a couple of steps can change almost entirely the category subject, the task is not as trivial as it may seem at first glance.

Once we obtain the final categories for our mission, we evaluate the correctness of our choices with the help of an interesting internet tool, called *Crowdflower*, and a separate small skilled user set. Through this two evaluation sources, we try to judge the correctness of our category/labels choices for the missions, and we consider eventual refinements.

1.1 Structure of the Thesis:

The rest of the Thesis is organized as follows:

Chapter 2 describes the Related Works. Here we will talk about preceding attempts at mission detection, entity linking and query/mission labelling, in order to give a basic background on what the attempts to tackle these problems have been.

Chapter 3 describes the problem we aim to face, the approach we choose to use, the explanation of our algorithm and of our work. We will show in detail all the phases that constitute our work, and explain them in detail.

In **chapter 4** we will explain how we tested our software. We will give details on the data-sets used, we will show the results obtained with the chosen parameters, and we will give information on the measure of evaluation and make a comparison with other existing approaches.

Chapter 5 is dedicated to the conclusions to our work. We will analyse the outcome of our work, and identify the reasons why it did or it didn't succeed. Further development ideas for our work will be given in the light of the obtained results.

Chapter 2

Related Works

In the following, we present preceding research activities concerning Search Mission Detection, Entity Linking, and Query and Cluster labelling. With these, we hope to give the reader a simple overview of the different techniques and of the older and more recent methodologies that have been presented in order to face these issues, which are at the center of our thesis.

2.1 Search mission detection

Query logs are widely considered a very rich source of knowledge and the interest around them is rapidly increasing within the Web mining research community. Briefly, query logs are collections of information about users' search activities and therefore they are a significant source of information that can be used to understand the way people search the Web and the actual intent behind the queries [62]. Query logs can be exploited for statistical analysis in order to obtain measures such as: query popularity, term popularity, average query length, etc.. With more complex analysis, it is also possible to obtain some "structured" data, such as search sessions, that is, temporal sequences of queries issued by users.

In digital library search systems, “sessions” are easy to identify: users log in, perform a single task and log out again, so login IDs can be used to identify them. Historically then, a session was simultaneously: (I) a set of queries aimed at satisfying a single information need. (II) a series of subsequent uninterrupted queries. (III) a period of contiguous time of navigation that could be spent submitting queries or examining results. On the Internet though, users almost never log in and out for each task. In addition, there’s the chance that an identifier such as an IP addresses or a cookie might be shared by more than one user, such as in a shared computer. Thus the term “session” has been split between these various meanings. In web search analysis, there have been a certain number of conflicting attempts to segment and define sessions, that rely either on a notion of similar context, topic, or temporal characteristics to perform the session separation.[35] Segmenting the query stream into sets of related information-seeking queries, i.e. logical sessions, has multiple applications: other than query recommendation, it can be used to help understand the relationship between queries, given the user intent, since logical sessions carry useful data for user profiling and personalization.

The earliest methods for session detection used a simple time-based approach. This was based on the intuition that two consecutive queries by the same user are likely to belong to the same session whenever the time gap between the two queries is smaller than some predefined limit. Different time gaps have been tried but it is not hard to understand that, because of the multitasking nature of the user interaction with web search engines, which means that the user switches between different search needs within a single user session, a single physical session will most likely include some query aimed at a different need. It has been seen in fact, that while searching the web, users don't usually follow a single search goal from beginning to end at a time, but instead they handle multiple search goals concurrently. This causes time-based approaches for session detection to be imprecise, or very hard to implement.[45]

Many works deal with identification of user search sessions boundaries. Previous papers on this topic, can be classified in 3 main groups based on the technique used: (I) Time-based, (II) Content-based, and (III) Mixed techniques, which usually combine (I) and (II).

2.1.1 Time-Based

Many time-based techniques have been proposed in the literature to detect significant search sessions. The strong point of these algorithms is their utter simplicity and ease of implementation. Indeed, these approaches are based on the (basically wrong) assumption that the time interval between two queries issued by a single user is the main factor determining a possible topic shift in a search activity.

We shall now see some examples of techniques exploiting this approach:

Silverstein et al. [61] first define the concept of session as a series of queries where each query is issued withing a 5 minute time window from the last one.

He and Göker do something similar in [31], where they study different possible time limits, ranging from 1 to 50 minutes, used to split user sessions.

Radlinski and Joachims [55] observe that users often perform chains of queries, which are sequences of queries based on a similar information needs. Their work consists in the presentation of a simple method to automatically detect query chains in query and click-through logs. The result of their studies shows that a 30-minute time-out is accurate over 90% of the times when they attempt to determine whether two consecutive queries belong to the same search session or not.

Jansen and Spink [33] make a comparison between nine search engine transaction logs based on session length, query length, query complexity, and content viewed. They also provide a new definition of session, that they call search episode, and describe it as the period of time between the first and the last recorded time stamp on the search engine server from a particular user on a single day. The session length may then vary from less than a minute to several hours, according to how the session is defined.

Spink et al. [63] investigate multitasking behaviours for user interactions with a search engine. The results, as we have stated earlier, show that multitasking is a common characteristic of Web searching. This result made it clear that it was not possible to rely only on time sessions in order to identify series of related queries.

2.1.2 Content-Based

Previous works regarding session detection, suggested to exploit the lexical content of queries in order to determine session boundaries. Those should correspond to possible topic shifts in the stream of issued queries, and, therefore, they should define a session boundary. Several search patterns have been proposed by means of lexical comparison based on different string similarity metrics (e.g., Levenshtein, Jaccard, etc.). However, approaches relying only on content features similarity, suffer the vocabulary mismatch problem, namely the existence of topically-related queries without any shared terms [45].

Shen et al. [60] tried to overcome this issue by comparing augmented representations of queries instead of the queries themselves. Each individual augmented query is built by concatenating the titles and the Web snippets for the 50 top results provided by a search engine for that specific query. The relatedness between query pairs is then computed by using the cosine similarity between the corresponding augmented queries.

Ozmutlu and Cavdur [51] define a mechanism aimed at identifying topic changes in the user's search behaviour thanks to a combination of time intervals and query content features. The validity of their approach is then tested using a genetic algorithm in order to learn the parameters of the topic identification task.

Gayo-Avello [24] Introduces a new technique for session boundary detection working on the basis of a geometric interpretation of both time gap and query content similarity, between consecutive query pairs. However, lexical features can't determine semantic relatedness, so this approach suffers the previously described vocabulary-mismatch problem.

For such cases multiple semantic features have been examined. A proposed idea similar to the one proposed by Shen et al. is to enrich the short query strings in order to get a longer, richer representation of the actual information need. In a very recent method [44] the well-known ESA framework [22] is used to check semantic similarity between query representations in a given collection (i.e., Wikipedia articles).

2.1.3 Mixed Heuristics

Boldi et al. [11] introduce the query-flow graph (QFG), a model to represent the data collected in WSE query logs. Intuitively, in the QFG, a directed edge from query q_i to a query q_j means that the two queries probably refer to the same information need.

Any path over the QFG may be seen as a searching behaviour, whose probability is given by the strength of the edges constituting the path. Boldi et al. then prove the correctness and the usefulness of their model in both session boundary detection and query recommendation.

Jones and Klinkner [35] present the first high-level, task-by-task analysis of user search behavior, by introducing the concept of "hierarchical search". They argue that it's possible to identify specific hierarchical units within a user's query stream. The units are in fact, search missions, that they subdivide into separate search goals. Jones and Klinkner define a search goal as an atomic information need resulting from one or more queries. On the other hand, they define a search mission as a set of topically-related information needs, that can result in one or more goals. The authors then focus on investigating how to learn a suitable binary classifier, which is aimed at detecting whether two queries belong to the same task or not.

Donato et al. [18] develop SearchPad, a novel Yahoo! Application [44]. This application adopts the same hierarchical structure of user search behavior, by introducing the concept of hierarchical search. Also, it is able to automatically identify search missions "on-the-fly", as the user interacts with a Web search engine. The software aims to help users keep track of the results they have already consulted. The novelty of this approach however is that it is automatically triggered only when the system decides, with a discrete level of confidence, that the user is actually undertaking a search mission.

Wen et al. [66] describe a query clustering method that exploits user logs in order to identify those web documents that users have selected when searching for a certain query. The similarity between queries is centred around four notions of query distance: the first notion is based on the keywords or phrases present in the query;

the second is based on string matching of keywords; the third is based on the clicked URLs; and the fourth on the distance that the clicked documents have, according to some pre-defined hierarchy.

Kotov et al. [39] discuss about methods to model and analyse user search behaviour that spans over multiple search sessions. The authors focus on two main problems:

- Given a user query, identify all related queries from previous sessions that the user has issued.
- Given a multi-query task for a user, predict whether the user will return to this task in the future.

Both problems are modelled within a classification framework using features of long-term user search behaviour and individual queries at different granularity.

Guo et al. [27] introduce the concept of intent-aware query similarity, that can be described as an approach for computing query pair similarity, taking into account the potential search intents behind user queries and then measuring query similarity for different intents using intent-aware representations. The authors then apply their approach to query recommendation, thereby suggesting diverse queries in a structured way to search users.

Hagen et al.[28] presents a new method for logical session detection, which follows the stepwise paradigm of the recent cascading method [29]. By improving each single step of the cascading method, and adding a new one that employs Linked Open Data analysis to detect possible semantic relatedness between queries, the new approach achieves a better efficiency and effectiveness in extensive experiments.

Lucchese et al.[46] try to understand how people search the web exploiting a novel task-by-task perspective, in contrast with the query-by-query fashion commonly used. They study how such an high-level view of users' search behaviours help realize a new task recommender system. The authors create the Task Relation Graph (TRG), which is a task-by-task search representation extracted from a query log. Briefly, the TRG can be described as a directed graph, whose nodes are the tasks performed by users, and the weights on the edges define the likelihood that, given a task performed by that user, it will also perform another directly connected task. They then proceed to show how a TRG may be exploited in order to generate novel task recommendations,which might be used to help the user achieve long-term missions.

2.2 Entity Linking

Entity linking is the task of linking a textual entity mention, possibly identified by a named entity recognizer in an unstructured text, and associated with the corresponding real world entity in the existing knowledge base. Note that an entity can be referred to by multiple mentions and a mention can refer to multiple entities.[59]

The typical information retrieval approach to indexing, clustering, classification and retrieval, is that based on the bag-of-words paradigm. The basic idea of the bag of words model is that of identifying a sequence of terms (also called spots) in the input text and to annotate them with unambiguous entities drawn from a catalogue. The choice of the catalogue is therefore a crucial element for the success of the approach. Recently, a good deal of works attempted to go beyond this paradigm with the goal of improving the search experience on structured, unstructured or semi-structured textual data. [21]

Bagga and Baldwin [8] use the bag of words model in order to represent the context of the entity mention. They tackle the problem of cross-document co-reference by comparing, the word vectors built from all the sentences containing mentions of the targeted entities for any pair of entities in two documents.

Ravin and Kazi [57] further refine the method of solving co-reference by measuring context similarity and by integrating it into Nominator [65], which was one of the first successful systems for named entity recognition and co-reference resolution.

Mann and Yarowsky [47] develop a method that clusters web search results for a set of ambiguous personal names by employing a rich collection of biographic information obtained via bootstrapped extraction patterns.

Raghavan et al. [56] explore the use of entity language models for tasks such as clustering people by profession and classifying politicians as liberal or conservative. In order to build the models, they identify the named entities in the TREC-8 corpus (a specific corpus developed for the Text Retrieval Conference in 1999) and compute the probability distributions over the words occurring within a certain distance of any instance labelled as person of the canonical surface form of 162 famous people.

Bunescu and Pasca [13] exploit a set of useful features derived from Wikipedia for entity detection and disambiguation. In their system they leverage the bag of words model to measure the cosine similarity between the context of the mention and the text of the Wikipedia article. In order to overcome the usual bag of words model deficiencies, they use a disambiguation SVM (Support Vector Machine) kernel which models the magnitude of each word-category correlation based on the Wikipedia taxonomy.

Generally speaking, the one essential step of entity linking is to define a similarity measure between the text surrounding the mention and the document associated with the entity. The methods using the bag-of-words model are fundamentally based on the feature of context similarity which clearly depends on the similarity measured by the co-occurrence statistics of terms between the text around the entity mention and the document associated with the entity. Because of this, they ignore the semantic relationship existing between concepts and cannot capture many existing semantic relations. The entity mention will then be mapped to the corresponding entity in the chosen knowledge base only if the compared texts contain some identical contextual terms.

The purpose of semantic relatedness measures is then to allow computers to reason about written text.

The main difference between the various techniques used in order to discover semantic relations is their knowledge base. This can be obtained from a manually created thesauri, even though most thesaurus based techniques are limited in the vocabulary for which they can provide a relatedness measure, since the structures they rely on must be manually built. In alternative, the knowledge base can be obtained from corpus-based approaches, obtaining background knowledge by performing statistical analysis of large untagged collections of documents.

The most successful and well known of these techniques is Latent Semantic Analysis (LSA)[42], which relies on the tendency of related words to appear in similar contexts. LSA offers the same vocabulary as the corpus upon which it is built, therefore it can only provide accurate judgements as long as the corpus is very large but, because of the corpus dimensions, the preprocessing effort required is significant.

Hence Wikipedia is now being used as a knowledge base thanks to its ever-expanding database and to the fact that it offers the best trade-off between a catalogue with a rigorous structure but with low coverage, and a large text collection with wide coverage but unstructured and noisy content.

Strube and Ponzetto [53] made a first attempt to compute measures of semantic relatedness using Wikipedia. Their approach, WikiRelate, takes some familiar techniques that had previously been applied to WordNet [2] and modify them to suit Wikipedia. Their most accurate approach is based on Leacock & Chodorow's [43] path-length measure, which takes into account the depth at which the concepts are found within WordNet. WikiRelate's implementation does pretty much the same, but uses Wikipedia's category graph instead. For any pair of words, WikiRelate attempts to find a pair of articles whose titles contain those words. Once they have been found it computes their relatedness based on the word-based similarity of the articles and on the distance between the articles' categories in Wikipedia's category graph. While the results are similar, in terms of accuracy, to thesaurus based techniques, the nature of Wikipedia, based on user collaboration, offers a much larger and constantly evolving vocabulary.

Cucerzan, in his work [16], proposes the first system able to recognize the global document-level topical coherence of the entities. The system tackles the entity linking problem by maximizing the agreement between the text of the mention document and the context of the Wikipedia entity, as well as the agreement among the categories associated in Wikipedia with the candidate entities. Cucerzan assumes that all entity mentions have a corresponding entity in the knowledge base. This assumption though clearly fails for a huge amount of mentions in practice.

Gabrilovich and Markovitch [23] achieve extremely accurate results with their newly developed technique, called Explicit Semantic Analysis (ESA). This technique is somewhat reminiscent of the vector space model that has been widely used in information retrieval, though, instead of comparing vectors of term weights in order to evaluate the similarity between queries and documents, their method estimates a relatedness score for any two documents by using the inverted index to build a vector for each document over Wikipedia articles, and by computing the cosine similarity between the two vectors. The name Explicit Semantic Analysis derives from the way these vectors are comprised of manually defined concepts, as opposed to the mathematically derived contexts used by Latent Semantic Analysis (LSA). The result of the process is a measure which can even approach the accuracy of humans. Additionally, this method also provides relatedness measures for any length of text because, unlike WikiRelate, there is no restriction for the input to be matched to article titles.

Mihalcea and Csomai [49] propose the Wikify system. This works in two separates stages: (I) Detection, which involves identifying the terms and phrases from which links are supposed to be made. Their most accurate approach to this stage of the process is based on link probabilities extracted from Wikipedia's articles. The detection approach is then to gather all n-grams found within a document and to keep those whose link probability exceeds a predefined threshold. (II) Disambiguation, in which the system ensures that the previously detected phrases link to the most appropriate article. In order to choose the most appropriate destination, Wikify's extracts features from the analysed sentence and from its surrounding words (the terms themselves and their parts of speech), and then compares these to the training examples obtained from the entire Wikipedia.

Milne and Witten, in their works [50][67] propose an approach that yields considerable improvements by exploiting three main ingredients:

1. The identification in the given text of a set C of context pages, namely pages that are exclusively linked by spots that are not ambiguous (therefore they only link to a single page/sense).
2. A relatedness measure between two pages *p1, p2* based on the overlap between the in-linking pages of $p1$ and $p2$ in Wikipedia.
3. A notion of coherence of a page p with the other context pages in C.

In [67], Milne and Witten show that the ESA approach we talked about earlier, has performances quite similar to that of their approach, with the latter being way cheaper to be computed since it doesn't require the whole Wikipedia textual content to be indexed.

Medelyan et al., in their work [48] re-use the Wikify approach for detecting significant terms, though their method differs in how it disambiguates terms. They are able to obtain results similar to the ones of Wikify, but much more cheaply, by balancing (I) the commonness of each sense and (II) how that sense relates to its surrounding context.

Chakrabarti et al. [40] further improves [67] by introducing two main novelties. The first one is the choice of scoring an annotation with two terms, one of which is local to the occurrence of the spot and the other one which is global to the text fragment. The second novelty of their approach is that they choose to model the annotation process as a search for the mapping that maximizes the sum of the (local + global) scores of the selected annotations, through the resolution of a quadratic assignment problem. They performed experiments over a manually annotated dataset, whose results showed that their approach yields a precision similar to that of Milne and Witten's approach but with a higher recall.

Dredze et al., in [19] proposes a solution that focuses on the classification framework to resolve entity linking. Their approach first develops a rich set of features which are based on the entity mention, on the source document and on the knowledge base entry, and then it uses a SVM ranker to obtain a score for each candidate entity.

Because the preceding works of **Chakrabarti** [40] and **Milne-Witten**[67] are unsuitable in a short text context, **Ferragina and Scaiella** design and implement Tagme [21], an annotation framework that, on-the-fly and with fairly high precision/recall, annotates short fragments of text with relevant hyper-links to Wikipedia articles. In Tagme, Ferragina and Scaiella use Wikipedia anchor texts as spots and the pages linked to them as their possible meanings. The Tagme framework solves ambiguity and polysemy in the many available anchor-page mappings by finding the collective agreement among them via newly developed scoring functions. Their tool obtains this by explicitly taking into account the sparseness of the anchors in the short input text. This is done exploiting the proper combination of the relatedness function among concepts proposed in [67] and several other statistics drawn from Wikipedia.

Shen et al. in their work [59] propose LINDEN, a framework aimed at linking named entities in a text with a unified knowledge base, constituted by Wikipedia and WordNet, by leveraging the semantic knowledge that can be derived from Wikipedia and the taxonomy of the knowledge base. It is assumed that the named entity recognition process has already been completed, and therefore the focus of their framework is only on the task of linking the detected named entity mention with the given knowledge base. To this aim they collect a dictionary about the surface forms of entities from four different sources in Wikipedia (i.e., entity pages, redirect pages, disambiguation pages and hyper-links in Wikipedia article), and record the count information for each of the target entities in the dictionary. Using this dictionary, they then generate a candidate entity list for each entity mention and try to include all the possible corresponding entities of that mention in the generated list. In the process they also leverage the count information in order to define the link probability for each candidate entity. Subsequently, they identify all the Wikipedia concepts present in the document where the entity mention appears.

2.3 Query labelling and Cluster labelling

2.3.1 Query labelling

Query labelling is the process of Natural Language Processing (NLP) which aims to identify a label, within a predetermined set, that best represents the domain of interest related to the information need behind a user-written query.

Nowadays a number of vertical search engines provide a specialized kind of information service whose aim is to satisfy a user's need based on a particular intent. In practice however, a user needs to identify his intent in advance so that he can decide which vertical search engine he has to choose in order to satisfy his intention. It would be convenient though, if a query intent identifier could be provided within a general search engine, so that it would be possible to accurately predict whether a query need concerns a particular type of vertical search engine, handling a certain domain. Also, since a user's query may not be limited to a single intent, it would be extremely helpful for the user to have a general search engine detect all the query intents, and then distribute the query to the appropriate vertical search engines and, finally, effectively organize the obtained results from the different vertical search engines to satisfy a user's information need.[7]

Past researches on query labelling can be divided (with some overlap) into two general categories: (I) Manual classification, which involves using human editors or very simple heuristics to obtain a set of static classification decisions. (II) Automatic classification which usually aims at using complex algorithmic methods, such as supervised machine learning, to cluster similar queries or to perform direct task or topic-based classification. [10]

Manual Query Classification

Many studies have attempted to Manually classify queries in terms of the informational actions of users, though we are only going to present a couple, because they are not of particular interest for us.

Broder, in his work [12] manually classifies a small set of queries into 3 separate task categories: transactional, navigational, and informational. To do it, he exploited a pop-up survey of AltaVista users followed by a manual inspection.

Beitzel et al. [9] use exact lookup of queries from AOL's Web search engine in a large database of semi-automatically categorized queries with the aim of analysing query behavior over time. In this approach they chose to use a set of 16 categories.

Automatic Query Classification

Other works on query classification are less focused on understanding large-scale properties of query logs and more focused on improving system effectiveness on each individual query. Unluckily, a lot of this work is proprietary and unpublished, though academic literature is growing constantly.

Kang and Kim [36] use query term distribution, mutual information, usage rate in anchor text, part-of-speech info, and also a combinations of the above, with the aim of automatically classify user queries into Broder's [12] taxonomy. This classification is then used to decide which of the query processing algorithms should be used for retrieval. Their system though achieves at best modest improvements in retrieval effectiveness.

Kang's and Kim's studies [36] in fact exhibit a substantial problem common to all query classification studies: achieving accurate classification, and in particular high recall, is extremely difficult for those queries that are inevitably short.

Gravano et al. approach the problem of query classification in [26] by using machine learning techniques in an automatic classifier in order to categorize queries according to geographical locality. In their work they sample small training, tuning, and testing sets of a few hundred queries each from a very large 2.4 million-query Excite log from December 1999. Following the analysis they discovered that data sparseness is a significant problem for their classifier. In particular they noticed that most queries are very short, and, specific to their geographical task. Also, it was observed that queries rarely contain key trigger words that are likely to identify whether or not a query is either "local" or "global" in geographical scope. They finally conclude that successful automatic classification has to necessarily draw on external sources of information to compensate for the small number of features that are usually present within a single query.

Shen et al. [58] use synonym-based classifiers to map the category hierarchies used by search engines to the ones proposed at the KDD Cup 2005 [5]. In particular, this mapping function is constructed by matching keywords between each category hierarchy. Shen et al. extend the keyword matching process so that it includes various grammatical forms. In their work they build three synonym-based classifiers by using the category hierarchies from Google, Looksmart, and an internal search engine based on Lemur [4], searching a crawl of the ODP[3] hierarchy (Open Directory Project). Also, they build a statistical classifier using SVM light [34]. The training data is collected by using the mappings that have been found for the synonym classifiers in order to find pages in the relevant ODP categories for each query. The terms from these pages, their snippets, titles, and category names are then stemmed and processed to remove all the stop-words and to be used to train the SVM classifier. They also use two ensemble classifiers, one using the 111-query training set to learn weights for each component classifier, and the other one giving equal weight to each component classifier.

Kardkovacs et al. [37] propose an approach called the Ferrety Algorithm for the KDDCUP 2005. Their approach is quite similar to that by Shen et al. [58] we have seen earlier. Kardkovacs et al. however, use the Looksmart and Zeal search engines to build their taxonomy mapping, an they continue by enriching the mapping process with stemming, stop-word removal, and the application of WordNet. These data is then used to train their own learning software and make classification decisions. In their work they also experiment with several different methods of feature selection.

Vogel et al. [64] took part at the KDDCUP 2005 too. Their approach relies on a Web directory to drive part of their classification. They build their category mapping by searching ODP with Google, and collecting the categories found in the top 100 retrieved documents. They also make use of spelling corrections and of alternate queries automatically suggested by Google, which help mitigate some of the most noisy queries in the KDD Cup query set. They then manually inspect all of the categories returned by Google, and map each directory node to up to three KDD Cup categories. Finally, they combine all available information and generate probability scores for each category via a logistic regression.

Hu et al. [32] develop a system that starts by recognizing a set of seed concepts. It then retrieves all those Wikipedia articles and categories that are supposed to be relevant to these concepts. The Markov random walk algorithm is then used to iteratively propagate intent from the seed examples into the Wikipedia graph and to assign an intent score to each Wikipedia concept. What they obtain after this process is an intent probability for each concept in Wikipedia, which clearly identifies the semantic boundary of the intent domain. They then continue by mapping each query into a Wikipedia representation. If the input query can be exactly mapped to a Wikipedia entity, they predict its intent based on its associated intent probability. Otherwise, they map the query to the most related Wikipedia concepts by using Explicit Semantic Analysis (ESA) [23], and then make the judgment based on the intent probabilities of mapped Wikipedia concepts, which overcome the semantic disambiguation issue. Even though the system shows good premises, the implementation remains extremely small-scale and limited to just three basic domains: travel; personal name; job.

Khoury et al. [38]. Implemented a system that exploits Wikipedia's encyclopaedia structure to classify queries step-by-step, using query words to identify titles, then choosing the articles based on these titles, and finally extracting the categories from the articles. At each step, the weights of the selected elements are computed based on those elements that have been found to be relevant in the previous step: a title weight will then depend on the words that caused the system to choose it, an article weight will depend on the titles ones, and a category weight will therefore depend on the articles ones. Unlike the work of Hu et al. [32], this system was a general classifier that could handle queries from any domain.

AlemZadeh et al. [6], which is also one of the co-authors of the preceding work, built a general classifier, but its fundamental principles differ from [38]. Instead of using titles and articles to pinpoint the categories in which to classify a query as was done in [38], the classifier shown in [6] used titles only to create a set of inexact initial categories for the query. Those categories are then used as starting points for the exploration of the Wikipedia's category graph. This exploration will then lead to the discovery of the best goal categories from a set of predetermined valid classification goals.

2.3.2 Cluster labelling

As we have seen before when we talked about query clustering a lot of research is being done on clustering algorithms and their applications in information retrieval. However, in comparison, very little work has been done on cluster labelling. A classic approach for cluster labelling is to apply statistical techniques for feature selection. This is done by identifying important terms in the text that should best represent the cluster topic. However, a list of significant keywords, or even phrases, are very likely to fail to provide a meaningful label for a set of documents and queries. In many cases, the suggested terms, even if related to each other, tend to represent different aspects of the topic underlying the cluster. In other cases, a good label may just not occur in the text.[14] User intervention is therefore required if we want to extract a proper label from the suggested terms so that we can successfully describe the cluster's topic. However, these human chosen labels are rarely indicated as "significant" by feature selection methods. For example, the JSD(Jensen-Shannon Divergence) method for feature selection identifies human labels as significant (appearing in the top five most important terms) for only 15% of the categories [14]. This result implies that human labels are not necessarily significant from a statistical perspective. There is therefore the need for a cluster labelling method that uses external resources.

One of the first systems that dealt with cluster labelling is the Scatter/Gather application developed by **Cutting et al.** [17]. In this system, in addition to the cluster's important terms, the titles of the documents that are closest to the cluster centroid are also considered for labelling, since usually titles are much more readable than a list of terms. On the web, the anchor text associated with the in-links to the cluster web pages can be considered as candidate labels since often anchors successfully describe the pages to which they link.

Radev et al. [54] generate a summary of multiple documents using cluster centroids produced by topic detection and tracking.

However, multi-document summaries are usually too long to be utilized as short comprehensive labels. Several labelling approaches attempt to enrich content-only terms by exploiting external resources for labelling.

Geraci et al. [25] use a modified version of the Kullback Leibler Divergence [41] to identify words that best represent the cluster's contents and are least representative of the contents of other clusters.

Carmel et al. [14] show us a cluster labelling solution that leverages the labelling task by Wikipedia. In their approach, candidate labels are extracted from Wikipedia pages in addition to the important terms that are extracted directly from the cluster content. Thus, selected Wikipedia categories "compete" with inner terms for serving as the cluster labels. In general, Wikipedia is a very successful resource for labelling, however inner terms should be considered for the cases when Wikipedia fails to cover the cluster content even though, thanks to its ever expanding nature, Wikipedia covers more subjects every second.

Chapter 3

Problem Statement and Algorithms

In this chapter we aim to show precisely how we deal with the mission labeling task, explaining in details all the definitions and algorithms necessary to describe our tool.

First we will formally state the problem including some definitions, and the various knowledge bases necessary in order to understand how our tool works. Next, we will describe in depth the actual implementation of the program.

3.1 Definitions

We shall start with some definitions that a reader might find useful in order to understand properly the different constituent parts of our project.

For search session, search goal and search mission, we adopted the definitions given by Rosie Jones and Kristina Klinkner in [35], which are:

DEFINITION 1. *A search session is all of a single user activity within a fixed time window.*

A session then, is defined by a chosen amount of time, during which a user performs one or more searches. No relevance is given to information needs or goals that the user might be trying to achieve.

DEFINITION 2. *A search goal is an atomic information need, resulting in one or more queries.*

A goal can be thought of as a set of related queries from the same user, all aimed at achieving the same specific task. It is not necessary for the queries to be contiguous in order to be considered part of the same search goal. Queries belonging to the same search goal might be interleaved with other queries having a different goal.

DEFINITION 3. *A search mission is a related set of information needs, resulting in one or more related goals.*

A mission can be considered an extended information need. Looking for different Hotels to stay while in Seville and looking for a guide for the city might belong to the same mission though not to the same goal, because while they have both the general aim of visiting the city of Seville, the goals are respectively to find an hotel and to find a guide for the city.

The resulting hierarchy is: a sessions may contain one or more different missions. Each mission might contain one or more goals. A goal may consist of multiple queries aimed at achieving the same specific task. Different missions, queries and goals can all be temporally interleaved.

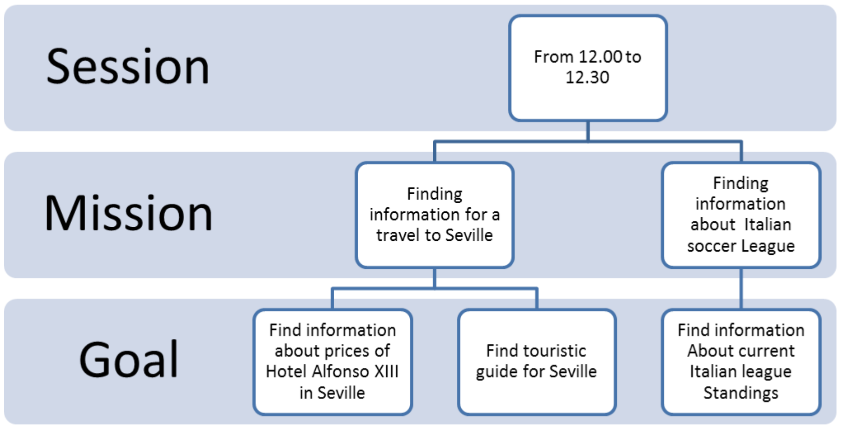

Session
From 12.00 to 12.30
Mission
Finding information for a travel to Seville
Finding information about Italian soccer League
Goal
Find information about prices of Hotel Alfonso XIII in Seville
Find touristic guide for Seville
Find information About current Italian league Standings

Figure 3.1: Scheme Representing the division between Session, Mission and Goal

Let us now define some basic elements and notation we will use to explain our work:

Let QNL be a query-navigation log containing all the data of our dataset, divided according to unique MissionIDs. Let $Q_i \in QNL$ be the list of Queries with MissionID i, and $N_i \in QNL$ the list of Navigation data associated to the same MissionID i. Navigation information can be described as the text extracted from the homepage relative to the website visited by the user during the search mission. Let q_i be a query belonging to the list Q_i of all queries with MissionID i. Also, let n_i be a single navigation data belonging to N_i.

We will now present some new fundamental concepts needed to understand our algorithm.

DEFINITION 4. *A Goal Category is a Category that is part of the set of Goal Categories that will be used by our tool as labels for the missions.*

In our project, as anticipated in the introduction, we chose to use Wikipedia categories to label the mission, and in particular, we used the set of 99 categories developed by AlemZadeh et Al. in [7]. The details about this set will be given in chapter 4, as they are of no interest to us for the algorithm description part.

Let's now explain what we mean by Wikipedia entity: when we talk about an entity, we are talking about a single specific **Article Page**, identified by an unambiguous PageID. An article has multiple properties, but the one we are interested in is that it belongs to one or more Wikipedia categories.

When talking about entities, we will refer only to article pages, and not to Wikipedia Categories, that will be referred to simply as Categories.

```
First Stage
Input:  $Q_i$ (list of queries belonging to mission $i$), $N_i$(Navigation
data belonging to mission $i$).

  1. $NE_i \leftarrow$ The list of entities extracted with Dexter from the
     Mission Navigation data.

  2. $QE_i \leftarrow$ The list of entities extracted with Dexter from the
     Mission Queries.

  3. $EL_i \leftarrow$ The complete list of entities extracted from both the
     queries and navigation data, with the scores assigned
     according to different factors, including whether they are
     entities extracted from $Q_i$ or $N_i$.  We also eliminate
     duplicate entities, after adjusting the score in order to
     consider the fact that the entity was found multiple times
     in the text.

Second Stage
Input:  $EL_i$, Wikipedia Database Dump(WDD)

  1. $CL_i \leftarrow$ we get the Scored Category list constituted by the
     categories extracted from our $EL_i$.  This Category List
     includes the Scores that each category will inherit from the
     entity it was extracted from.

  2. $CL_i \leftarrow$ We do a further step into refining our scoring by
     searching the category set for duplicates of categories, and
     use this information to adjust the scores accordingly.  Also
     we eliminate duplicate categories.

  3. $SCS_i \leftarrow$ After all the steps previously taken in order to
     obtain a final scoring, we prune all the categories that do
     not meet a certain score threshold, defined according to the
     highest category score for that particular mission.

Third Stage
Input:  $SCS_i$, WCG(Wikipedia Category Graph), $GC$ (Goal Categories)

  1. $DIST(SCS_i, GC) \leftarrow$ We compute the distance between each of the
     Categories in $SCS_i$ and their closest Goal Category, by
     exploiting the Breadth First Search algorithm on the WCG
     extracted from the WDD we used.

  2. $FCS_i \leftarrow$ We now have the List of Goal Categories that were
     reached from the categories in $SCS_i$.  We take the Goal
     Categories with the path length equal to that of the
     shortest path found.  If there is more than one, we consider
     as a higher ranked Category those that were reached the
     highest number of times by the categories of the $SCS_i$.

  3. We return the Final Category Set and use it as label for our
     mission.
```

Table 3.1: Structure of the three stages of our labeling algorithm

3.2 Algorithm

Mission log creation

Being our Algorithm aimed at Search Mission labelling, it is necessary to have as input a search Mission log. There exist multiple methods to develop a Mission log such as the one we take as input for our labelling tool. As we have shown in the Related Works chapter, this process can be done both via supervised as well as unsupervised techniques. In both cases the objective is to identify and group all those queries submitted by the same user that aim at a similar information need. Once all the related queries have been identified through a manual process or by exploiting some similarity measure, we can consider them all as part of the same mission. We therefore give them the same MissionID. In the mission log we will consider, if a web-page has been visited as result of a query, the homepage of the website to which the visited web-page belongs, is recorded in the Mission log too.

3.2.1 General Algorithm Description

We will now describe the algorithm developed to classify the search missions contained in a *QNL* as the one we defined earlier.

- The **First Stage** of the process aims to extract the entities from the queries and from the navigation data of a given mission. Following it performs the first score computing step.

- The **Second Stage** extracts the starting set of categories from the obtained entities, then further refines the scores in order to give the final scoring of our categories. A pruning process eliminates the categories with the lowest scores before the graph exploration.

- The **Third Stage** begins with the set of Starting Categories, and computes the distances from each one to the closest objective category. After a further ranking, the best categories are chosen as mission labels.

In 3.1 we can see the general summary of our algorithm.

3.2.2 First Stage

We begin the first stage by taking in input the two starting data structures fundamental for our labelling goal: the structure containing all the queries belonging to the mission we are analysing (Q_i), and the structure containing all the navigation data related to the same mission (N_i).

Entity extraction

Now, our process starts by extracting Wikipedia entities from these two data structures. If there is no navigation data available for a mission, the navigation data part analysis will be ignored and every correlated process will be skipped. As database we used the freely available Wikipedia dump from October 2013. These dumps are made available by the Wikimedia Foundation Project.

Extracting entities from a text, and solving possible ambiguities, is an extensively documented job often referred to as **Entity Linking**, with multiple different approaches attempted in the last 10 years, as shown in Chapter 2 of this thesis. Unluckily most of these approaches are not publicly available.

We chose to rely on a new entity linking tool called **Dexter** [15], developed by Ceccarelli et al. at the university of Pisa, in collaboration with Ca' Foscari. We shall now explain how entity linking generally works and how Dexter works.

Dexter

Given a plain text, the Entity Linking task consists in identifying small **fragments of text** (also called spots or mentions) referring to any entity that is listed in a given knowledge base. Usually the task is performed in three steps:

1. **Spotting:** a set of candidate mentions is detected in the input document, and for each mention a list of candidate entities is retrieved.
2. **Disambiguation:** for each spot associated with more than one candidate entity, a single one is chosen to be linked to the spot.
3. **Ranking:** the list of detected entities is ranked according to some score or policy.

The framework developed by Ceccarelli, called Dexter, is a standalone program developed in JAVA, designed to work through a set of Rest APIs or through a web interface.

The main components of the software are depicted in Figure 3.2. The spot repository contains all the anchors used in Wikipedia for intra-linking the articles. For each spot, the spot index contains the link probability.

DEFINITION 5. *Link Probability is the probability for a spot to be a link, estimated as the occurrences of the text as anchor in the Knowledge Base, divided the occurrences as pure text.*

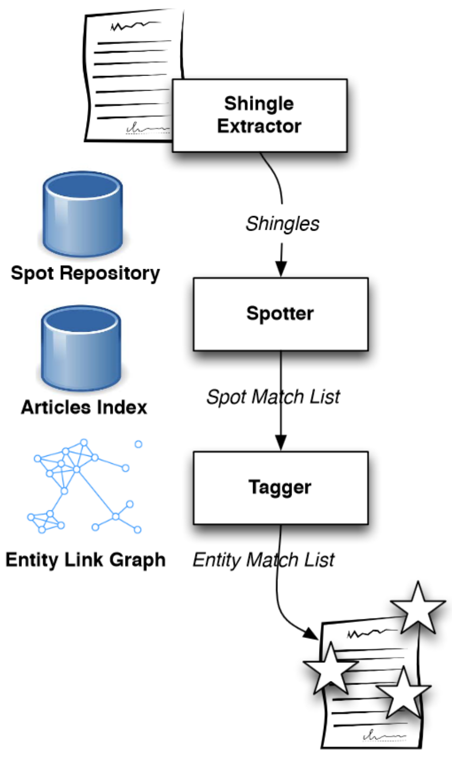


Figure 3.2: Scheme Representing how Dexter Works

The article index is a multi-field index of the article, built by using Lucene [1], while the entity graph stores the connections among the entities.

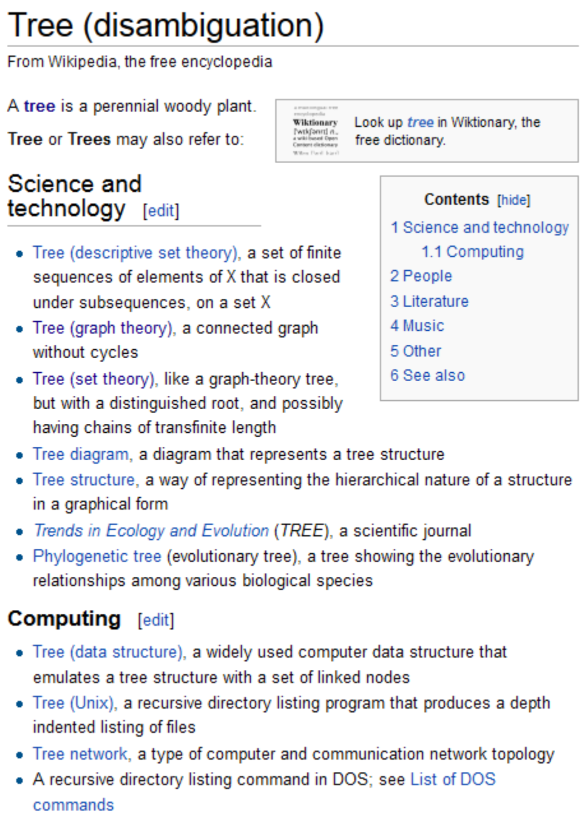
Dexter begins by using the **Shingle Extractor** to produce a list of possible mentions in the given text. The shingle extractor produces all the possible n-grams of terms, where n ranges from 1 to 6.

It should also be remembered that each entity can be referred to by more than one name. This property is called **name variation**, and it means that an entity can be mentioned in different ways such as full name, aliases, acronyms and misspellings. As an Example, there are multiple ways to search for John Fitzgerald Kennedy in Wikipedia. You can either look for *"John F. Kennedy"*,*"John Fitzgerald*

Kennedy", *"JFK"*, *"John Kennedy"*, or *"Jack Kennedy"*, and Wikipedia will always redirect you to the page of the famous president of the United States of America. **The spotter** has now the job of identifying all the entities in the text, by searching for any fragment of text that matches any mention. A name mention is a set of 1 or more words identifying a possible Wikipedia entity.

The Tagger then takes as input the list of spot matches produced by the spotter and selects the best entity for each spot, performing the necessary disambiguations.[15]

Disambiguation



Tree (disambiguation)

From Wikipedia, the free encyclopedia

A **tree** is a perennial woody plant.

Tree or **Trees** may also refer to:

Look up *tree* in Wiktionary, the free dictionary.

Contents [hide]

1 Science and technology
 1.1 Computing
2 People
3 Literature
4 Music
5 Other
6 See also

Science and technology [edit]

- Tree (descriptive set theory), a set of finite sequences of elements of X that is closed under subsequences, on a set X
- Tree (graph theory), a connected graph without cycles
- Tree (set theory), like a graph-theory tree, but with a distinguished root, and possibly having chains of transfinite length
- Tree diagram, a diagram that represents a tree structure
- Tree structure, a way of representing the hierarchical nature of a structure in a graphical form
- *Trends in Ecology and Evolution* (*TREE*), a scientific journal
- Phylogenetic tree (evolutionary tree), a tree showing the evolutionary relationships among various biological species

Computing [edit]

- Tree (data structure), a widely used computer data structure that emulates a tree structure with a set of linked nodes
- Tree (Unix), a recursive directory listing program that produces a depth indented listing of files
- Tree network, a type of computer and communication network topology
- A recursive directory listing command in DOS; see List of DOS commands

Figure 3.3: The disambiguation Page for Tree

Disambiguation of mentions is always a problematic step if we consider that when a single mention is identified, it might link to multiple Wikipedia entities that might be referred by the same name. Tree for example, has many possible meanings and, amongst others, it can refer to the plant, to the graph construct or it might be an acronym for Trends in Ecology and Evolution (see Figure 3.3).

In the Dexter implementation we used, the disambiguation phase was based on **TAGME** [21]We will now briefly describe how Tagme works, in order to give the necessary background on Dexter's disambiguation phase works.

Lets give some definition first: a Wikipedia page is denoted as p. Because of polysemy and variant names, we denote by $Pg(a)$ the set of all Wikipedia pages linked by a (a text anchor).

DEFINITION 6. *$freq(a)$ is the number of times the text a occurs in Wikipedia (as an anchor or not).*

DEFINITION 7. *$link(a)$ denotes the number of times the text a occurs as an anchor in Wikipedia (so $link(a) \leq freq(a)$).*

DEFINITION 8. *$Pr(p|a)$ denotes the prior-probability (a.k.a. commonness) that an occurrence of an anchor a points to $p \in Pg(a)$.*

Less formally, the commonness of a sense is defined as the number of times that the page describing that sense of the word is used as a destination in Wikipedia.

The annotation of an anchor a with some page $p \in Pg(a)$ is denoted by $a \mapsto p$. Often a has multiple senses, thus $|Pg(a)| > 1$, so the disambiguation process consist in selecting one of the possible senses of a from $Pg(a)$. Obviously not all occurrences of the spot a should be considered as text anchors, so Ferragina and Scaiella introduce a fake page NA that is used in the annotation $a \mapsto NA$ in order to denote that an occurrence of anchor a has been pruned.

Let A_T be the set of all anchors occurring in an input text T, Tagme tries to disambiguate each anchor $a \in A_T$ by computing a score for each possible sense p_a of a

(hence $p_a \in Pg(a)$) via a new notion of *"collective agreement"* between pa and the possible senses of all other anchors detected in T.

To this aim, Tagme deploys a new voting scheme that computes for each other anchor $b \in A_T \setminus \{a\}$ its vote to the annotation $a \mapsto p_a$. Given that b may have more than one sense (i.e. $|Pg(b)| > 1$) we compute this vote as the average *relatedness* between each sense p_b of the anchor b and the sense p_a we wish to associate to the anchor a.[21]

Relatedness has been previously defined by Milne and Witten in [50] as a measure $rel(p1, p2)$ between two pages $p1, p2$ based on the overlap between their in-linking pages in Wikipedia;

Formally, the relatedness measure is:

DEFINITION 9. $relatedness(a,b) = \frac{log(max(|A|,|B|)) - log(|A| \bigcap |B|)}{log(|W|) - log(min(|A|,|B|))}$

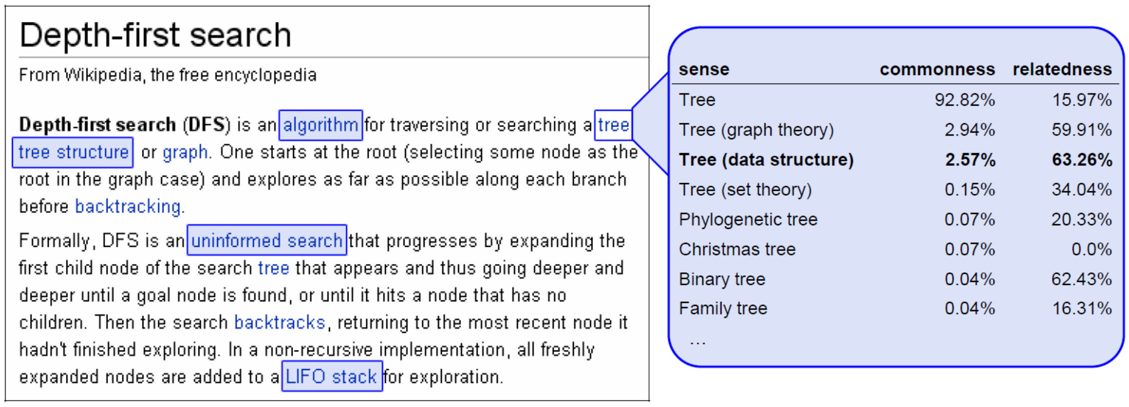
Where a and b are the two articles of interest, A and B are the sets of all articles that link to a and b respectively, and W is the set of all articles in Wikipedia. The relatedness of a candidate sense is the weighted average of its relatedness to each context article.[50]

Since not all possible senses of b have the same (statistical) significance, we weight the contribution of p_b by means of its commonness $Pr(p_b|b)$.

Hence the formula is:

DEFINITION 10. $vote_b(p_a) = \frac{\sum_{p_b \in Pg(b)} rel(p_b, p_a) \cdot Pr(P_b|b)}{|Pg(b)|}$

We notice that if b is un-ambiguous, $Pr(p_b|b) = 1$ and $|Pg(b)| = 1$, so we have $vote_b(p_a) = rel(p_b, p_a)$ and therefore we fully deploy the unique senses of the un-ambiguous anchors. If b is polysemous though, only the senses p_b related to p_a will mainly affect $vote_b(p_a)$ because of the use of the relatedness score $rel(p_b, p_a)$. The total score $rel_a(p_a)$ for the ”goodness” of the annotation $a \mapsto p_a$ is computed as the sum of the votes given to it by all other anchors b detected in the input text.


Depth-first search

From Wikipedia, the free encyclopedia

Depth-first search (**DFS**) is an algorithm for traversing or searching a tree, tree structure, or graph. One starts at the root (selecting some node as the root in the graph case) and explores as far as possible along each branch before backtracking.

Formally, DFS is an uninformed search that progresses by expanding the first child node of the search tree that appears and thus going deeper and deeper until a goal node is found, or until it hits a node that has no children. Then the search backtracks, returning to the most recent node it hadn't finished exploring. In a non-recursive implementation, all freshly expanded nodes are added to a LIFO stack for exploration.

sense	commonness	relatedness
Tree	92.82%	15.97%
Tree (graph theory)	2.94%	59.91%
Tree (data structure)	**2.57%**	**63.26%**
Tree (set theory)	0.15%	34.04%
Phylogenetic tree	0.07%	20.33%
Christmas tree	0.07%	0.0%
Binary tree	0.04%	62.43%
Family tree	0.04%	16.31%
...		

Figure 3.4: An Example of how disambiguation works, using Milne and Witten relatedness and prior-probability(commonness) measures

To disambiguate a, and therefore select its best annotation $a \mapsto p_a$, Ferragina and Scaiella choose to use a newly developed method called Disambiguation by Threshold (DT). Disambiguation by Threshold recognizes a roughness in the value of $rel_a(p_a)$ among all $p_a \in Pg(a)$, so it computes the top-ε best senses p' in $Pg(a)$ according to their $rel_a(p')$, and then annotates a with the sense that obtains the highest commonness among them. DT discard from the above computation all senses $p \in Pg(a)$ whose prior probability $Pr(p|a) < \tau$, where τ is set as explained in [21].

After the disambiguation phase, Dexter's Tagger outputs an entity match list, with the position in the original text of each annotated entity and a confidence score.

First Scoring Step

Once we have our Entity list, still divided according to whether they were extracted from N_i or from Q_i, we proceed by assigning to each entity their score.

We will call QE_i and NE_i the entity lists extracted from Q_i and N_i respectively by Dexter.

In the first step of the scoring process we take the link probability (given by Dexter) from each entity identified in the given text, and we use it as a base score for our entities.

We chose Link Probability as base score for the entities, because it is a good measure of how likely it is that the particular entity we found, was written because it was significant for the mission. As an example, ”the” has an extremely low link-probability, because even though it is present a huge amount of times in Wikipedia pages, it is almost never used as an anchor for its Wikipedia article page, and it is obviously almost never significant in a sentence.

After assigning the base score, we proceed to increase that score, based on whether the mission entities are present in both queries and navigation data.

For each qe_i (a single query entity), in QE_i, we check if an identical entity ne_i (a single navigation data entity), is present in NE_i too. Two entities are said to be identical if they have the same unique PageID defined in the Wikipedia database. When two identical entities are found, the navigation data entity score will be multiplied by a factor f, which depends on the total number of entities present in NE_i. This will be done for each entity qe_i in QE_i. This means that if there are k identical qe_i entities in QE_i, the score of the entity ne_i identical to qe_i, will be:

DEFINITION 11. $score(ne_i) = linkprobability(ne_i) + (linkprobability(qe_i) \cdot f \cdot k)$

The resulting entities in NE_i with the newly computed scores, are then added to a new entity list we will denote with EL_i.

This scoring choice is performed because while websites are dispersive, even though they might contain meaningful information, the queries should represent what the user is actually looking for. Therefore, if an entity is present in both the Query and in the Navigation data, it will probably mean that the entity is highly significant for the user's search mission, while if it is only present in the query, it might mean that the entity found in QE_i was a simple mistake done by the user or by Dexter, which could have disambiguated a mention incorrectly because of the limited context given by the queries.

This might not be the case though, so if no entity in NE_i is equal to an entity qe_i, qe_i will be added to EL_i, and it will have a score equal to it's $linkprobability$ multiplied by a factor v. The factor v depends on the total number of entities in NE_i. This score increase is performed so that, in case NE_i contains a large amount of entities, these do not completely overwhelm all those entities in QE_i that are not also present in NE_i.

This query score multiplication is made on the base that the supplied queries are supposed to be meaningful. If no other identical entity is found in QE_i, the entity will probably obtain a final score lower than the ones of other entities found in both QE_i or NE_i. On the other hand, if the entity is present multiple times in QE_i, it should mean that the entity has an a high relevance for the mission, and the performed score multiplication will contribute to make it relevant in the mission labelling process.

Second Scoring Step and First Pruning

In the last part of this stage, we check EL_i for occurrences of entities with PageIDs identical to each other. We then choose one instance of the identical entities, el_i, and after adding the values of all the other duplicates to el_i, the other identical entities are removed. The resulting EL_i will then be a set with no duplicates. From this set, we chose to only take the highest score entities in order to keep computation time reasonable and to avoid analysing low score entities, because they would most likely be misleading for the mission labelling process, and would anyway end up being pruned in the next stage.

3.2.3 Second Stage

Category Extraction and Third Scoring step

In the Second Stage of our tool, we start by extracting the Categories to which the entities in EL_i belong. We do this by exploiting the JWPL APIs, a set of APIs developed by the Wikimedia foundation to allow programmers to interact with the Wikipedia Database Dumps. We now have a list of categories relative to the mission i that we will call CL_i. Each of the categories in CL_i will inherit the score from the entity it was extracted from. We identify with cl_i a single category of the Category List CL_i so:

DEFINITION 12. $basescore(cl_i) = score(el_i)$

Where el_i is the entity from which cl_i is extracted, and $score(el_i)$ is the score that each entity in EL_i has at the end of the first stage. We then proceed to a further scoring step.

First of all let's consider two separate, but not necessarily different, categories cla_i and clb_i in CL_i, then the score of each category cla_i will be:

DEFINITION 13. $score(cla_i) = \sum_{\forall cla_i = clb_i} basescore(clb_i) \cdot j$

where j is the number of times a category clb_i identical to cla_i appears in CL_i.

This step is performed to increase the score of those identical categories extracted from multiple different entities.

This can help us reveal underlying relationships between entities that would have otherwise passed unnoticed. In fact it might probably be that a category extracted from multiple different entities is relevant for the search mission need. Considering that the English Wikipedia has more than 400.000 categories, it is not common to find multiple identical categories casually within a 30 or less categories list. For this reason, the score of a category is equal to the sum of the score of its identical categories in CL_i, multiplied by the number of the identical categories instances in CL_i which we called j. Also, the set of the categories of an entity often include some category that might be considered irrelevant, because only marginally connected to the topic that the entity handles. As an example, the entity *"Harford County"* belongs to the categories *"Maryland counties; Harford County - Maryland; 1773 establishments in Maryland; Populated places established in 1773"*. It is clear that the most significant categories for this entity can be considered *"Maryland Counties"* and *"Harford County-Maryland"*, while the other two categories are probably less significant for that entity. If we also have, in the same mission, the entity *"Harford County Sheriff's Office"*, from it we will extract the categories *" Harford County, Maryland; Sheriffs' offices of Maryland; 1774 establishments in Maryland"*. Having *"Harford County, Maryland"* in common, this category value will be increased, giving it a higher relevance in the labelling process.

At the end of this step we obtain a further refined Category List, that will be passed on to the following stage.

Low scores Pruning

With the Category List obtained from the preceding step (CL_i), we proceed to perform a further pruning. It can happen that some categories have a score that is too low compared to that of the ”best” categories. Because after this step we will only take the categories with the highest score and then stop using the scores to rank them, we need to do a final pruning, and eliminate all those that might be considered irrelevant.

We remove from our CL_i all those categories that have a Score which is less than $1/k$ that of the highest score category in CL_i, where k is a factor that can be changed according to whether we want or not to consider lower score categories into our set. This step is necessary to eliminate those categories that have score too low to prove significant for the mission. Our Category Set is now ready.

3.2.4 Third Stage

Wikipedia Category Graph Exploration

The Category set we now have, will be called Starting Category Set, or SCS_i. The categories contained in it are all the categories we will use as starting points in the Category Graph in order to reach our Goal Categories.

To do this we need to exploit the Wikipedia Category Graph (WCG). The WCG represents the pseudo Taxonomy of the Category Graph. It is not a taxonomy or a tree, because its structure contains short-cuts, isolated leaves with no parents and cycles. Also a category might be linked by more than a single other category. We can see an example of a small part of the graph in Figure 3.5. Nonetheless the Category Graph gives us the chance to explore it in order to find our Goal Categories. The Goal Category Set is a Set of predefined Categories, built to cover most if not any possible subject that might be the goal of a web search mission. Obviously, these categories will be quite general in order to cover all of the possible subjects of a search mission, otherwise it would require a huge amount of more specific categories to cover them.

Our Algorithm travels this graph exclusively from a category to its parents. We only explore the graph ”upwards” because the goal categories are general enough to make it almost if not completely impossible(judging by all our tests) for an entity to belong to a category on an upper level of the graph, i.e. the goal category *”Science”* has as parent categories only *”Main topic classifications”* and *”Academic disciplines”*, which are obviously two ”container” categories. Our objective now is to explore the WCG, searching, for each Starting Category in SCS_i, for the Goal Category that can be reached in the minimum number of steps. The algorithm starts the exploration of the graph from a Starting Category taken from our SCS_i, and it continues travelling the graph until the closest Goal Category is found. The closeness of a Goal Category is determined by the number of $Children \rightarrow Parent$ steps it takes to reach it starting from our Starting Category. At that point we can suppose that we have found the Goal Category that best represents our starting category.

DFS

Two methods have been tried for graph exploration while developing the tool. Depth First Search and Breadth First Search. In our DFS algorithm, we recursively explore the graph in his depth, starting from the categories in our SCS_i. Once any of the Goal Categories is found, the number of steps to reach it is recorded, and the algorithm proceeds recursively to explore the graph, looking for any Goal Category requiring less steps to be reached. We should always consider that there are a huge number of paths that can lead to any of the Goal Categories, so we need to examine all the available paths from our starting category to find the shortest one. To improve efficiency, a technique to avoid visiting a category if it has already been visited in a preceding step has been implemented. This also helps avoiding cyclic category repetitions that are sometimes present in the WCG.

Because of the complex structure of the graph though, some visits can take a great number of $Children \rightarrow Parent$ steps. This can happen because there are exploration paths that do not lead to a Goal Category, and so the graph visiting process keeps on travelling the WCG, visiting Categories that are used to group other categories, and that have no real meaning for us, such as *"Main Topic Classification"* or *"Articles"*. Though, because there is not a clear way to define the depth of a category in the WCG, because of cycles, short-cuts and non-linked categories, it was impossible to define a depth at which the recursion had to stop. As general limit, we chose 20 as maximum number of $Children \rightarrow Parent$ steps before stopping the exploration. It is safe to say that no category will need more then 20 steps to reach the closest Goal Category. In fact, categories require an average of 3.05 steps to reach the closest Goal Categories.

The DFS will eventually come to an end when the closest Goal Category is reached and no category is left to be visited. Considering that each entity has an average of 2.68 categories each, and that each category usually has even more parent categories, it is easy to see that a DFS execution would most likely require a huge amount of time. Exploring the WCG might in fact require $|SCS_i| \cdot 2.68^{nsteps}$ where $|SCS_i|$ is the cardinality of the Starting Category Set, 2.68 is a low estimate of the average number of parent for each category and $nsteps$ is the average number of steps that the DFS algorithm takes to reach a Goal Category during the whole execution. Because of the limit we set, $nsteps$ will be less or equal to 20.

BFS

Because of the complications found in the DFS we chose to use the Breadth First Search for our tool. With a BFS algorithm, we can obtain quick results. By visiting all the categories present at each step before proceeding to the next $Children \to Parent$ step, we have a way faster exploration. That is because, as said before, the goal categories can be reached in an average of 3.05 steps, giving us an estimated required number of steps of: $|SCS_i| \cdot 2.68^{3.05}$. This way we avoid exploring uselessly a large part of the WCG with the chance of not finding any relevant or close enough Goal Category like it can happen with the DFS. The main advantage of this approach over the DFS is then the greatly reduced computation time, and for this reason we chose it over DFS. Also, in case of a future use of these tool for jobs that require a quick/real-time execution, it is obvious that DFS would be unusable.

Final Pruning and end of the Process

Now, after finding the set of Goal Categories reached by the categories in our SCS_i, we create a new set by pruning all those categories that took more steps than the shortest path found to reach. We chose not to consider any Goal Category reached through a path longer than the shortest one because, as we can see in Figure 3.5, a single step can change the subject radically. In the graph of Figure 3.5 it only takes 3 steps from Computing, to reach *"Historical eras"*, and then *"History"* (on the right side of the picture), which is a completely different topic, even if, by exploring the connection that binds them, it can be seen that there is a logical link. At the same time though, with 3 steps on the left, the subject doesn't change that drastically with categories such as *"Technology Systems"* or *"Technology"*. This "Quick Subject Change" property that the Wikipedia Category Graph has, is one of the biggest problems our tool has to face.

We will now have a Final Category Set containing all those categories that required a number of steps equal to the shortest path found.

From this Category Set, we take the 3 categories, or less if we don't have enough categories, with the path length equal to the shortest path. If two categories have the same path length, the goal category with the most number of duplicates in SCS_i after the BFS will be chosen. Eventual duplicates in this group of 3 categories will not be substituted by the next closest goal category in order to avoid including irrelevant categories.

We now have our Mission Labels.

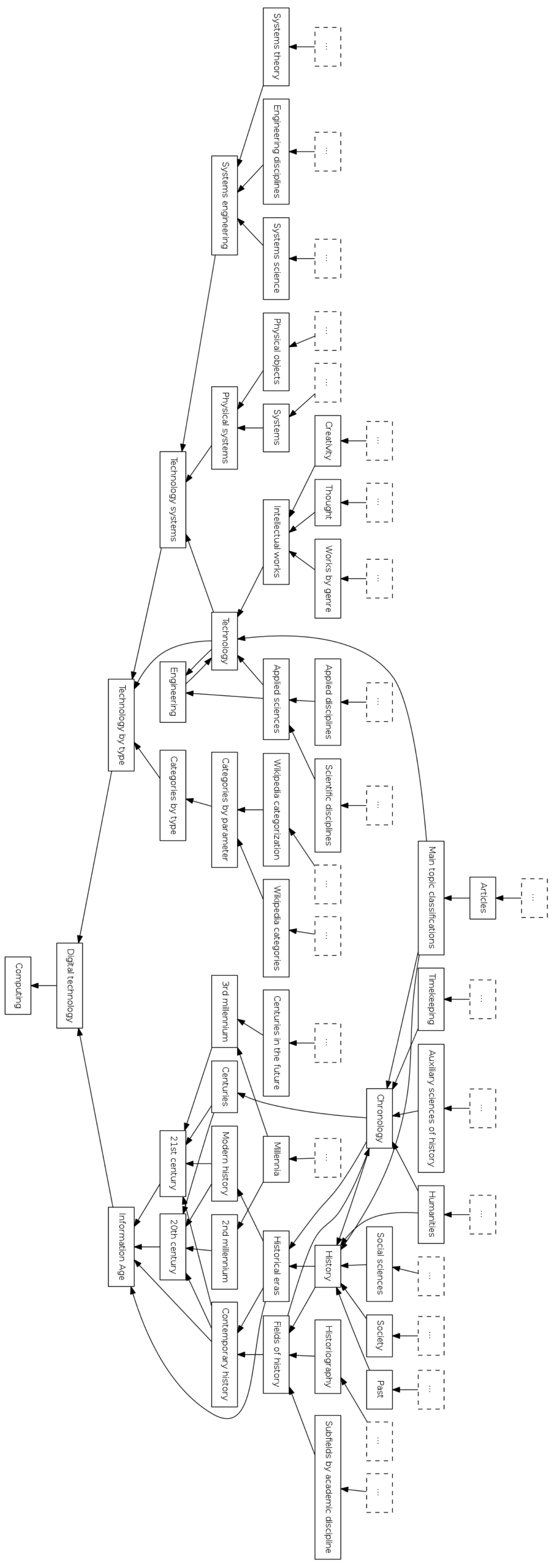


Figure 3.5: Upward Exploration of Computing category in WCG

Chapter 4

Content of the project

After explaining the algorithm we developed for our Automatic Labelling tool, we will show the implementation choices, the actual dataset we used, the structure of the Wikipedia Category Graph, and the chosen test parameters and evaluation methodologies.

4.1 Implementation Choices and Data

4.1.1 Mission Dataset

Being the objective of our thesis the labelling of a set of user search missions, we first had to choose a search mission log. To our knowledge there is no other very large search mission log, other than the one developed by Hagen et al. at the university of Weimar [28]. Their log is based on the Gayo-Avello's query log corpus [24], which consist of 11484 queries from 215 users sampled from the 2006 AOL query log [52] with respect to representativeness of typical querying behaviour (ratio of repeated queries, click through rate, etc.). In Gayo-Avello's corpus, a single human annotator subdivided the sample into 4040 sessions.

There are however some clear problems with Gayo-Avello's corupus:

- The notion that he gives of session, focuses only on series of consecutive queries aimed at the same information need, without taking into account the chance that different sessions with the same information need might be related.
- The query submission time and the relative clicks are not given within the corpus, and therefore it might be necessary to reconstruct them from the AOL log.
- Almost half of users submitted less then 4 queries at different times, making session detection not really useful in those cases. With that fer queries we don't have enough information to judge whether the queries belong to the same mission or not.
- From time to time, the ordering of queries in the Gayo-Avello's corpus is not coherent to the one of the AOL query log (and it is not possible to understand the order without timestamps).
- Some queries are just left out of the corpus for no apparent logical reason.

Because of all the issues we described, Hagen et al. couldn't use the corpus for session detection as it was, therefore they took the Gayo-Avello's data-set as basis, removed the few empty queries and those that only contained a URL (probably because the user mistakenly inserted the URL in the search space instead of inserting it in the address bar), and also removed the users that submitted less then 4 queries in total as it would be hard to obtain a correct or significant mission. The query sample obtained, has 8840 queries from 127 different users.

The next step was for two human annotators to divide this sample of queries into 2881 logical sessions and 1378 missions. When in doubt, there was a discussion in order to reach a consensus on which query belonged to which mission. The corpus they obtained is the one, slightly altered, we used in order to develop and test our tool. The official name of Hagen's corpus is Websis Search Mission Corpus 2012 (Websis-SMC-12). The statistical data for the obtained corpus are:

- Each user conducts an average of 10.85 missions of 6.42 queries each.
- Each mission contains on average 2.09 logical sessions.
- Missions and sessions that are interrupted by a different session or mission at some point, are picked up again later in 1505 cases by 93 users.
- 76.3% of missions are ended within one day.
- The longest non-trivial mission runs for 88 days.

The Websis-SMC-12 corpus is structured like this: there is a line for each interaction with the user; the interactions are ordered by time (something that wasn't always true in Gayo-Avello's corpus); each line contains a user ID, a query string, a timestamp and the rank and the homepage of the clicked result in case there is a click interaction. As a result of the logical session and mission identification work they did, to the basic data were added a mission ID for each interaction, and, even if not always, there is a manual annotation that aims to describe the intent of the mission.

In Table 4.1 we show an example of the dataset entries. Because the table is too large to fit in a single page, we reduced it by eliminating the rank of the clicked domain, the mission id and the manual annotation. The missions are still identifiable because each mission of the same user is separated by an empty row, while different users, and consequently also different missions, are separated by a line between the UserIDs.

UserID	Query	QueryTime	ClickURL
——			
38534	maryland state police	2006-03-01 23:48:15	http://www.mdsp.org
38534	harford county sheriff	2006-03-01 23:55:13	http://www.harfordsheriff.org
38534	aberdeen police department in maryland	2006-03-01 23:59:35	http://www.aberdeen-md.org
38534	aberdeen police department in maryland	2006-03-01 23:59:35	http://www.aberdeen-md.org
38534	goody	2006-03-20 16:19:46	
38534	goody	2006-03-20 16:19:47	
38534	goody	2006-03-20 16:20:08	
38534	goody	2006-03-20 16:20:09	
38534	google	2006-03-20 16:26:54	
——			
932882	accountsummary	2006-03-07 06:23:01	
932882	americanexpress accountsummary	2006-03-07 06:23:31	
932882	americanexpress accountsummary	2006-03-07 06:23:36	
932882	americanexpress account summary	2006-03-07 06:23:52	http://apply-credit-card.us
932882	americanexpress	2006-03-07 06:24:51	
932882	american greetomgs	2006-03-13 11:49:39	
——			

Table 4.1: Hagen's dataset Example

In order to make our tool for automatic mission labelling work, we didn't need some of the elements that Hagen et al. included in their corpus. The time-stamp, the number of the clicked URL in the search result page, and the eventual annotation they gave to the mission were removed because not useful to our intent. An identifier made by a combination of user and mission id, the query text and the general domain of the clicked URL were all that we needed. In order to unequivocally identify each mission, we marginally modified the corpus. The user and mission IDs in the Hagen's corpus have been combined in order to obtain a single mission ID that can identify the set of interactions that are part of a single mission, regardless of which user submitted it (even if it is easy to understand by looking only at the first part of the mission id). For each line, after the mission ID, there is the query submitted by the user during that particular interaction with the web search engine, and finally, there is the URL clicked by the user within that same interaction. There can be different interactions within the same mission ids, that will be shown as different lines with the same Mission ID. An example of our modified query-log can be seen in Table 4.2

4.1.2 Goal Category Set and WCG Structure

After Building our starting data-set, we reasoned about the kind of categories we could use for the labelling task. Looking at different preceding attempts at labelling queries or sessions, we saw that recently almost all of the focus is on Wikipedia. Using Wikipedia does seem logic, because there is no other publicly available source of universal knowledge that is so vast, detailed and linked as Wikipedia is. Wikipedia gives free access to it's database dumps, and also there are different groups that are developing APIs that give the necessary tools to handle these databases. In particular, because we chose to develop our tool in JAVA language, we relied on JWPL (Java Wikipedia Library), a set of APIs that allow for handling Wikipedia dumps through methods interacting with not only the single data, but also with the underlying struc-

USER_MISSION_ID	QUERY	CLICK_URL
38534_1	maryland state police	http://www.mdsp.org
38534_1	harford county sheriff	http://www.harfordsheriff.org
38534_1	aberdeen police department in maryland	http://www.aberdeen-md.org http://www.aberdeen-md.org
38534_2	goody	
38534_2	goody	
38534_2	goody	
38534_2	goody	
38534_3	google	
932882_1	accountsummary	
932882_1	americanexpress accountsummary	
932882_1	americanexpress accountsummary	
932882_1	americanexpress account summary	http://apply-credit-card.us
932882_1	americanexpress	
932882_2	american greetomgs	

Table 4.2: Modified Hagen Dataset Example

ture of the Wikipedia database [68].

After researching previous uses of the Wikipedia database for labelling or categorization, we understood that to use Wikipedia categories as labels for our missions, we had to choose a limited set of categories in order to restrict their number to a reasonable amount, such that it is large enough to comprehend all of the possible subjects that a user might touch while using a web search engine while not being too general. Also, we didn't want a set of extremely specific categories because that would render the labelling task useless, considering that what would come out of the process would have probably been so specific, that every mission would have had a different mission label (something that is not that unlikely if we consider that in the English Wikipedia there are now more than 400.000 categories).

We have gone looking for preceding attempts at developing a set of Wikipedia categories as the one we were looking for, and found a set of categories chosen by AlemZadeh et al. that fitted perfectly our scope. In their work, AlemZadeh et al.[7] developed a set of 99 Wikipedia categories they used to classify single queries. The 99 categories set is an adaptation of the 67 categories that were given to all participants during the KDDCUP 2005 [5] about which we talked about in the related works when describing the preceding labelling attempts. What AlemZadeh et al. did, is finding in Wikipedia the categories corresponding to those of the KDDCUP. They did it not only because the KDDCUP categories was in fact a well developed set of categories that well represented most if not any of the possible subject that a user search mission can touch, but also because that gave them a way to compare their results to those of the groups who competed in the KDDCUP 2005.

In Tables 4.3, 4.4 and 4.5 we can see the 67 KDDCUP categories and the corresponding 99 Wikipedia category set built by Hagen et al.

KDDCUP Categories	Kharray's Wikipedia Category
Computers\Hardware	Computer Hardware
Computers\Internet and 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\Others	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	Entertainmnet
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

Table 4.3: Alemzadeh 99 Categories With The 67 KDDCUP2005 Categories, Part 1

KDDCUP Categories	Kharray's Wikipedia Category
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
	Furnishing
	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
Living\Travel & Vacation	Travel
	Holidays
Living\Other	Personal Life
Online Community\Chat & Instant Messaging	On-line chat
	Instant messaging

Table 4.4: Alemzadeh 99 Categories With The 67 KDDCUP2005 Categories, Part 2

KDDCUP Categories	Kharray's Wikipedia Category
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 & Discount	Sales promotion
	Bargaining theory
Shopping\ Other	Distribution, retailing, and wholesale
Sports\ American Football	American football
Sports\ Auto Racing	Auto racing
Sports\ Baseball	Baseball
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

Table 4.5: Alemzadeh 99 Categories With The 67 KDDCUP2005 Categories, Part 3

We can see that while the KDDCUP categories are divided into 2 levels, the Wikipedia ones do not give any information about the level to which they belong. This is not casual, but it is caused by the particular way the Wikipedia Category Graph (WCG) is built.

In Wikipedia, the categories are organized in a structure that slightly resembles a taxonomy, though it does have cycles and shortcuts as we can see in Figure 3.5, therefore it doesn't comply with all the requirements of a taxonomy graph, as explained in chapter 3, which makes it harder for us to properly analyse it. Each category can have any number of subcategories, which includes the possibility that it has none (this usually happens for specific categories which will be the leaves of the graph). A category C usually belongs to another category S (so C is a subcategory of S) when C handles some subject that is a specification of the more general subject handled in S. As an example, the category Automobiles has many subcategories, amongst which there is Automotive industry, which can be easily identified as a specification of the Automobiles subject.

Each article in Wikipedia has at least 1 category to which it belongs, while there is no restriction to the maximum number of categories the article belongs to. As in 2010, the average number of categories per page was 2,68. Also, the average distance between any 2 categories was 5.5568 steps, which can give an idea of how much connected the graph is, considering that there are more than 400.000 categories. [20]

As an example of the degree to which a category expands in just a few steps, we can see a graphical representation of the first 2 steps from the parent category ”Mathematics” to its descendants and their subsequent descendants in Figure 4.1 [30].

Talking about the single categories, a category C contains a list of its subcategories, of the pages that belong to the category C and of the categories to which C belongs to (we call them parent categories). Sometimes, but not always, a really short description of the subject of the category is present in the page of the category. An example can be seen in Figure 4.2

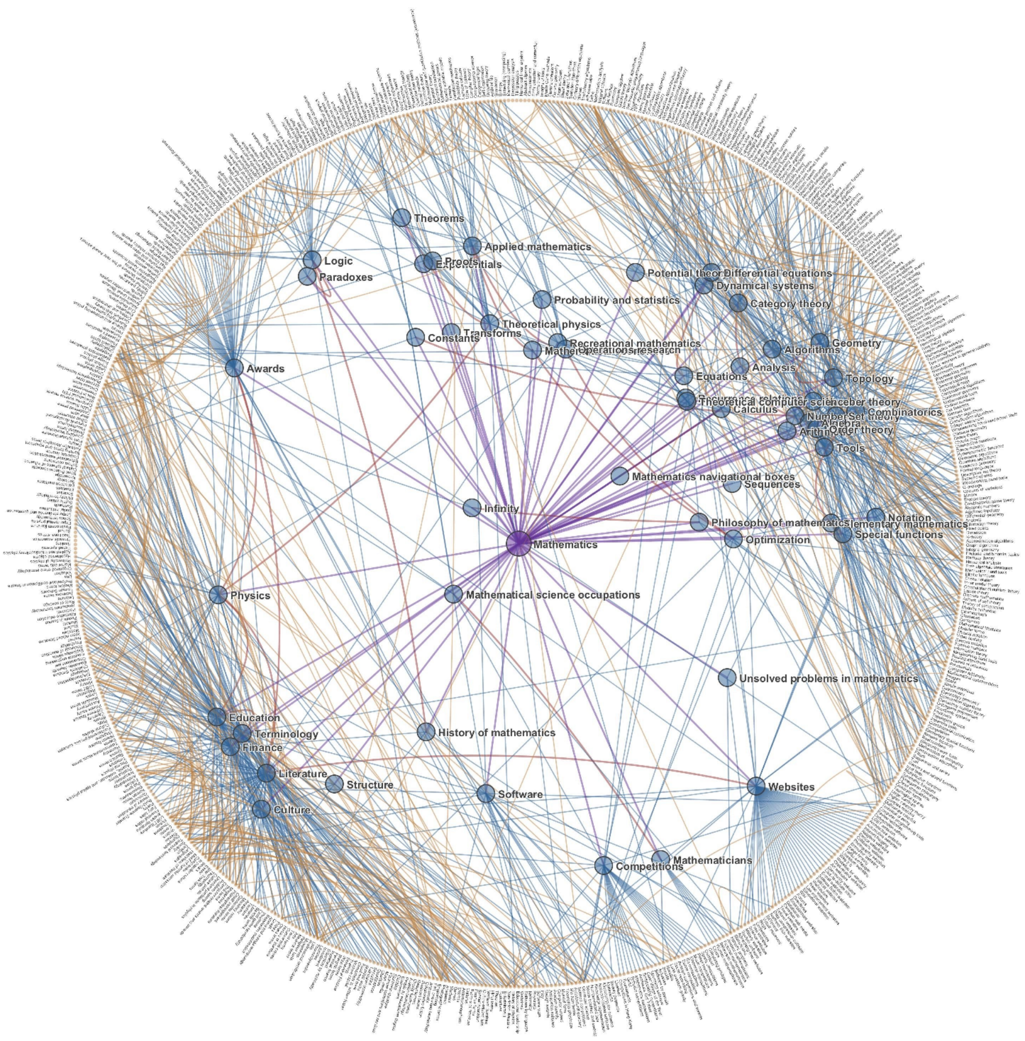
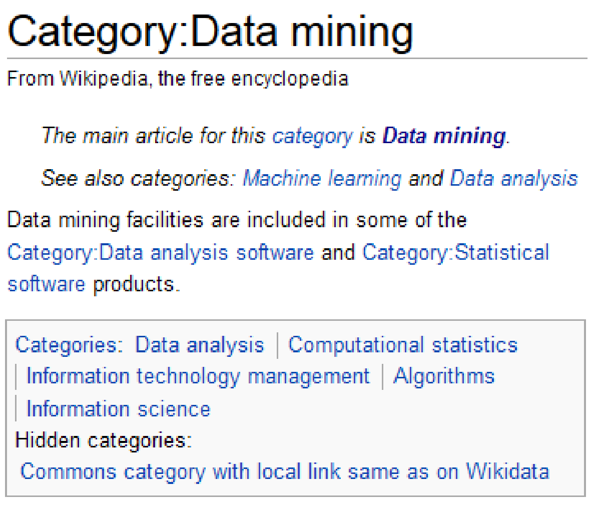


Figure 4.1: A "Clusterball" representing 2 steps of Mathematics descendants

In each category is also present a list of the Hidden Categories to which the category belongs to. Hidden categories are categories that are used by editors or administrators for maintenance reasons, i.e. to group pages that need to be modified, or categories that describe a very particular property, like *"Commons category with page title same as on Wikidata"*.

Each article must belong to at least one category that is not an hidden category. During our study of the Wikipedia category graph, we saw that from time to time, there are inconsistencies.

For example the category *"Chrysler"* doesn't belong to any category, while it would seem obvious that it should belong to some automotive related category. At the same time *"Fiat"*, another car company that can be considered similar at an encyclopedic level, belongs to different categories: *"Motor vehicle manufacturers of Italy; Defunct aircraft manufacturers of Italy; Multinational companies; headquartered in Italy; Conglomerate companies of Italy"*. We came to wonder why is there such a discrepancy between two categories that look so similar, and the only possible answer is that Wikipedia, because of its collaborative and social nature, is subject to mistakes. What is even more strange is that in the 4 months it passed since we discovered this mistake, no correction was made. Unluckily for us, these mistakes influence our tool. For example, in a very well studied mission labelling task, we had to label different queries talking about some Chrysler models, and it is obvious that if we had the chance to start our task by analysing the Chrysler category, we could have had a big advantage and obviously better precision. These cases though are quite rare


Category:Data mining

From Wikipedia, the free encyclopedia

The main article for this category is ***Data mining***.

See also categories: Machine learning and Data analysis

Data mining facilities are included in some of the Category:Data analysis software and Category:Statistical software products.

Categories: Data analysis | Computational statistics | Information technology management | Algorithms | Information science

Hidden categories:
Commons category with local link same as on Wikidata

Figure 4.2: Excerpt of the Data Mining Category page

and also don't compromise completely the effectiveness of the tool, as you have seen in the section dedicated to the explanation of our algorithm.

4.1.3 Query data and Navigation data

In order to perform our algorithm, we need the entities extracted from the queries and the entities extracted from the navigation data. While extracting entities from a specific mission query is trivial, as we just need to group the text of the queries in a single data structure, it is not as easy with the navigation data.

As we have shown earlier, the starting dataset contains the queries and the web address of the domain relative to the visited web page. Obviously the address of the domain is not much use to us, because no significant information can be extracted from it considering that it is extremely unlikely for a web address to correspond to a Wikipedia entity.

We then tried to extract some content from the websites that the user visited during the mission. In order to do that, we downloaded the web-page from the indicated domain, and extracts all the text contained in the paragraphs, denoted with $< p >$ in HTML. As *User agent* information given during the download of the html, we chose *Mozilla 5.0 (X11; Ubuntu; Linux x86-64; rv:25.0)*, in order to ensure compatibility with most websites.

Despite this, different issues surfaced while trying to download the web-pages. Sometimes the web page had been removed, other times it was inaccessible or required credential. Other websites were either down for maintenance or had been moved to another domain. It should in fact be considered that the AOL Query-log from which the dataset is extracted, was built in 2006, so it is to be expected that some Websites do not exist any more. It also happened that some websites built with

Microsoft Frontpage for some reason could not be analysed and ended up returning no text. Common exceptions were SSLException, IOException, ConnectException, SocketTimeoutException and others too.

Initially, we considered extracting the whole text of the web-page, including links and menus, but that resulted in a huge amount of non significant and misleading text, mostly menus, links, headers and footers or sidebars. Those mostly contained non relevant or generic words, and in many cases they also contained links to Twitter, Facebook or other social networks, that are normally used to share the content of the page. The actual significant text was then drowned in a huge amount of non-significant text. After trying to extract different parts of the HTML such as $< meta >$, $< body >$, and other parts that we thought could have been relevant, we chose to extract the only part of text that seemed to be relevant or related to the content of the website most of the times. The content of paragraphs in the HTML was therefore extracted from all the visited domains of the mission. The only problem with this solution was that Dexter only accepts strings of a limited length, so we made the choice to split the text extracted from a single website if it was longer than 3000 characters. Those text parts were later ”annotated” (annotation is the process of entity linking performed by Dexter) separately.

By looking at the text extracted from the HTML paragraphs, we have a majority of good text, but many times there are parts of it that still refer to copyrights, contacts, social networks and other parts of a common website that have no connection with its content. This is due to the non-consistent nature of websites, that don’t have strict structure rules that web designers need to follow. This makes the extraction of only the significant text from an HTML by means of an automatic system to be very hard if not impossible. Also, having the actual visited web pages would have helped our efforts immensely, because most home-pages contain images, menus, lists of content, but no large amount of real written content describing the subject of the web page. For a possible future use of this tool we would advice exploiting a dataset containing the actual visited pages, and not the corresponding homepage. Also, it would be advisable to explicitly state the keywords describing the content of the website inside the meta-tag keywords in the HTML, which is something that, by our experience, is almost never done.

4.2 Testing and Evaluation

4.2.1 Evaluation Tools

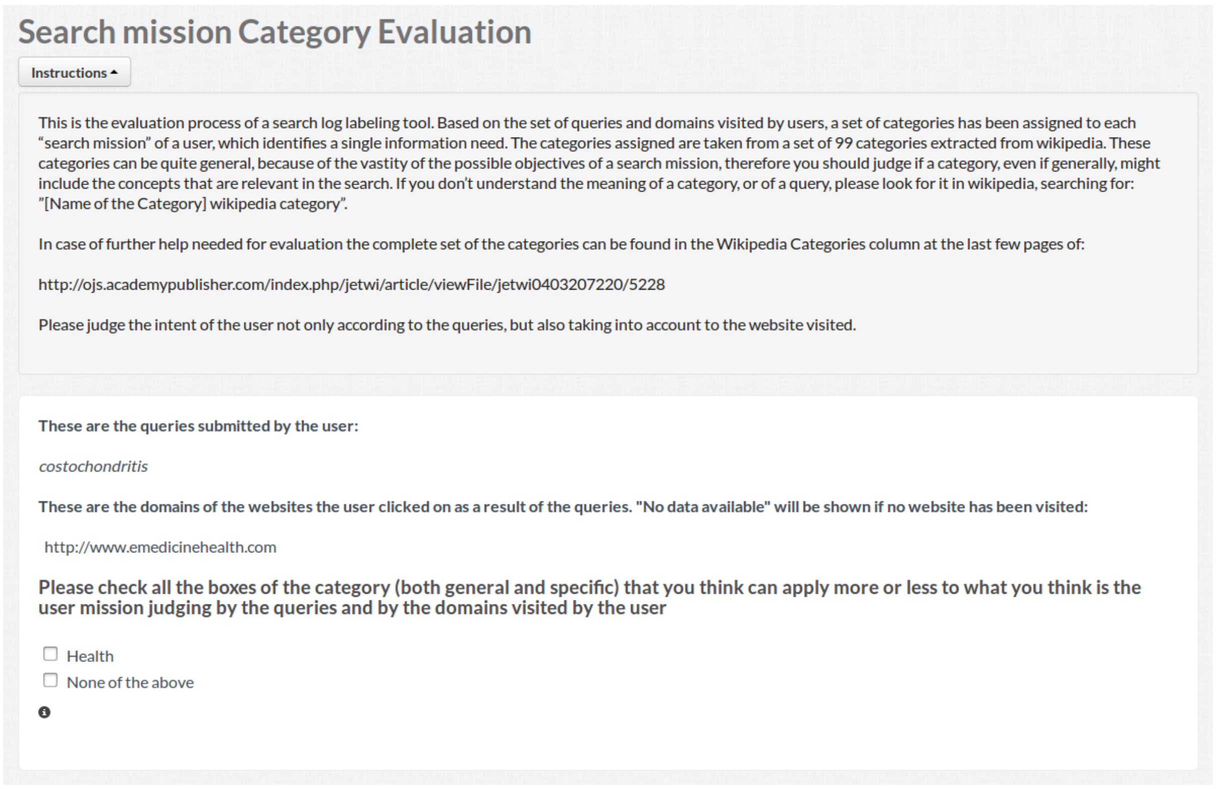
For the evaluation of our project, we chose to adopt a user study approach, by using two different sources of evaluation.

The first source of evaluation, is Crowdflower. Crowdflower is a company that uses crowdsourcing techniques to provide solutions that aim at processing large amounts of data. In practice, Crowdflower gives the tools and resources necessary to perform evaluation tasks that are usually performed by a single individual or by a small group. To do this, they exploit a network of evaluators that are paid per single evaluation. The amount of money is set by the client designing the task between 0.01 dollars and 1 dollar per single evaluation. The cost should be set according to the evaluation task complexity, so that there is a fair trade-off between time required for the evaluation, and the given pay.

The job design can either be done through a graphical interface, or by writing the necessary code in CML (a proprietary version of HTML) and javascript/CSS. Because the graphical interface did not provide us with the necessary tools we needed to build our job, we provided our personalized CML with the related javascript necessary to make the Job work.

We asked the contributors to evaluate which of the categories assigned by our automatic labelling tool to a single mission were deemed appropriate.

In order to give the evaluators the necessary information, we provide the queries and the URLs of the visited websites. Providing the user with text extracted from the mission URLs has been considered, though it proved impossible because of the length of the extracted text.



Search mission Category Evaluation

This is the evaluation process of a search log labeling tool. Based on the set of queries and domains visited by users, a set of categories has been assigned to each “search mission” of a user, which identifies a single information need. The categories assigned are taken from a set of 99 categories extracted from wikipedia. These categories can be quite general, because of the vastity of the possible objectives of a search mission, therefore you should judge if a category, even if generally, might include the concepts that are relevant in the search. If you don’t understand the meaning of a category, or of a query, please look for it in wikipedia, searching for: ”[Name of the Category] wikipedia category”.

In case of further help needed for evaluation the complete set of the categories can be found in the Wikipedia Categories column at the last few pages of:

http://ojs.academypublisher.com/index.php/jetwi/article/viewFile/jetwi0403207220/5228

Please judge the intent of the user not only according to the queries, but also taking into account to the website visited.

These are the queries submitted by the user:

costochondritis

These are the domains of the websites the user clicked on as a result of the queries. "No data available" will be shown if no website has been visited:

http://www.emedicinehealth.com

Please check all the boxes of the category (both general and specific) that you think can apply more or less to what you think is the user mission judging by the queries and by the domains visited by the user

☐ Health
☐ None of the above

Figure 4.3: An example of the Crowdflower Evaluation Form

The evaluators also had the possibility of consulting the list of the 99 goal categories in case of doubts. Each unit of our test was evaluated by 5 separate evaluators in order to give a decent amount of fairness in the obtained results.

The second source of evaluation is a small set of four skilled users. They were properly trained and instructed to analyse carefully the search mission and the 99 Goal Categories before giving a judgement on the mission labels.

The aim of this small skilled users set, is to give us the chance of checking if Crowdflower contributors are careful with their decisions, or if they don't pay much attention and don't put any effort in trying to understand what the mission is about.

4.2.2 Testing

For the proper tests, we explored different combination of values, changing in particular the factor f defined in Definition 11, and the factor v described as the multiplication factor of an entity qe_i not found in NE_i. As said in Chapter 3, those factors depend on the number of entities found in NE_i. In particular we chose v equals to $\frac{k \cdot |NE_i|}{10}$, and f equals to $\frac{j \cdot |NE_i|}{10}$.

In the testing process we won't refer to f and v because the number of entities in NE_i changes at each mission, but instead we will change the values of j and k. j will also be called *"Navigation enhancement Value"*. and k will be called *"Query enhancement value"*. This last value can be described as the factor for which the score of an entity will be multiplied, for every 10 entities in NE_i.

Another parameter that will be changed in the tests, is the score limit we chose in order to prune the categories with a score lower then a certain level compared to the highest score category. This will be simply referred to as "score limit".

As a last setting, we chose to change the number of entities that Dexter returns from the text we pass. We doubled the standard number of entities so that it now is 10 entities for each piece of text of a maximum of 3000 characters. This allowed us to consider more entities at each time and to reach a better precision. Higher values were tried, but the computation time required increased accordingly, while this choice did not always return an improvement. Therefore we preferred to keep computation times reasonable, while not compromising precision.

We first ran a base test to be used as reference for the improvements brought by our software, and then we performed different tests with different values to check whether a certain factor combination increases precision or not.

In the base test, we run our software without performing any operation other than the extraction of the entities, the extraction of the categories and the breadth first search. No factor f or v, no increase in score according to the presence of multiple identical instances of an entity, and no check on whether two or more identical categories were extracted from different entities has been performed. Only in the last part, the choice between two Goal Categories reached in the same number of steps, was made based on whether one of the two categories was found the most number of times by the set of categories in SCS_i while exploring the graph.

Because Mission Labelling has not been attempted much until now, we don't have a reference score or a precise definition of whether our classification is right or wrong and as a consequence, our judgement is focused only on the obtained precision.

The test dataset was a 100 subsequent set. These were chosen randomly, only eliminating those missions that were either non understandable even for us, or simply wrong, such as users writing the website address in the search field.

Baseline results evaluation

We start by evaluating the base tests we talked about earlier. No improvement has been made, and the results given by this test are given strictly by Dexter and, as told earlier, by the final choice of one shortest path Goal Category over another, based on which category was found the most number of times by the set of categories in SCS_i while exploring the graph.

It must be noted though, that even in this case, the Entity Linking part of the process can exploit the context of the mission while entity linking the text, and therefore obtain a more precise disambiguation over the same process performed on a single query.

As we can see from the test, the 34.4% of our labels was found to be correct, by the Crowdflower evaluators, while our skilled testers found the 34.74% of the labels to be correct.

It should also be noted that The Crowdflower agreement level was only 70.8%. Other data is that the average number of Categories belonging to the starting category sets of each mission is 28.75, while the average number of labels assigned to each mission is 1.52, with an average path length of 1.49 steps.

In these tests therefore, eve though we will present both the results of the Crowdflower evaluators and the skilled evaluators test, we will tend to give a higher relevance on the skilled evaluators. This was decided also by an in-depth analysis of the Crowdflower votes. By looking at what is probably the most clear mission of our dataset which is nothing more then the query ”internet” repeated 9 times, we can see that even though 4 out of 5 people deemed the label ”Internet” given by our tool, to be correct, 1 evaluator found it to be wrong. Considering that we can safely say that the label ”Internet” is the most appropriate label for the mission, we can evince that Crowdflower evaluators sometimes just click at random. This is probably due to the fact that they get paid per-answer, and therefore speed is the main factor determining income (even if we think that it probably required more work not to click on the Internet label check-box rather than clicking on it).

The labels in this base example are derived from just those categories with the highest score which, because there is no other improvement applied, are the categories derived from the entity with the highest *Link Probability*. Also, in case there were two Goal Categories being reached with the same path length by two Starting categories in SCS_i, it was chosen the Goal Category that was found the most number of times by all the categories in SCS_i. With only these refinements, it was clear that the results wouldn't be very accurate.

Correct labels come mostly from really clear and unambiguous mission such as the one presented before, consisting in the query ”internet”, repeated 9 times, with no website address visited by the user as a result of the query. Other missions that are found to be correctly labelled are composed by queries that are extremely clear, unambiguous and correctly structured, with few visited websites, such as the mission including the queries ”accountsummary; americanexpress accountsummary;americanexpress account summary ;americanexpress”, and with a single visited website ”http://apply-credit-card.us”. It is clear that achieving a correct labelling when the queries are correct, consistent, and the visited websites represent the actual subject of the query, is not hard. While clear and unambiguous queries are easy to label even without exploiting our tool completely, the problems start to arise when we start to handle ”harder” missions.

Even when trying to label a mission that have very clear queries such as ”camaro; history of the camaro; 1966 camaro; 1967 camaro photos”, that clearly refer to the famous American muscle car, the tool has problems identifying the correct label. In fact the labels that were assigned to this mission were ”Science; Technology”. This happens because in this base example, we don't exploit the steps of the tool that are used to emphasize the importance of concepts that are present both in the queries and in the webpage text. Because of this, the tool is not able to understand that the term

”camaro”, is more important than all the terms found in the many clicked websites. In fact, the six clicked web pages, even if most of them handle the ”Car” subject, in this case they only contribute by adding many non relevant entities. Also, we can see that one of the visited web-page is Wikipedia, but because in the mission log they chose to only record the home-pages relative to the actual web-page clicked by the user, we can’t exploit all the precious information that the ”Camaro” Wikipedia page would have held. This, as anticipated, is one of the main problems we found when implementing our software.

Obviously there are some missions that are very ambiguous or have a large set of visited websites that do not handle the query subject directly. Those missions will always be hard, or even almost impossible to label properly with our tool (as we will see later on), but we aim to have a great increase in precision over our base example, which is not too much different from a basic query labelling tool.

Multiple Tests with different parameters

After defining a baseline for our labelling evaluation tool, we tried exploiting all the techniques implemented in the software, in order to reach the best precision possible in the tests. To do this we started by trying different combinations of parameters, adjusting them according to the results given.

We tried multiple different combinations, in order to understand which directions to undertake when fine tuning the parameters, and we then adjusted our parameters step by step, until satisfactory results were reached.

First Test

First we started by analysing how our tool performed with a basic *"Navigation enhancement Value"* of 1 and a *"Query enhancement value"* of 2. The score limit was set at 1, restricting the categories in the Starting Category Set to those with a score equal to the highest score in SCS_i. This way we exploit our tool functionalities but without giving a very high relevance to the entities found only in the queries. This was a good way to understand how our process would behave even with no optimization.

The results of the Crowdflower evaluators showed a precision of 62.8% with a level of agreement of 73.61%. We should promptly point out that these data is not to be taken in consideration. The results given by the Crowdflower evaluators were analysed and were found to be quite random. This is probably due to the evaluators used, which change every time an evaluation is done. This causes the results to oscillate according to the evaluators' skills. This is easy to see when we analyse the data returned by our skilled user set, giving us a precision of 47.625%. There is too much of a difference from the skilled evaluators and the Crowdflower evaluators, therefore we will consider the Crowdflower data to be wrong, this choice is is later confirmed by all the other tests. We will anyway keep it in consideration when analysing the whole Crowdflower data, but we mark them as unreliable. The 47.625% of precision that our skilled evaluators found in this execution of our tool though is a clear sign that the processes put in act by our software increased the precision from the baseline by a great measure, showing how our tool is useful for labelling. More data extracted from this test show that the average number of Categories belonging to the starting category sets is 3.96, while the average number of labels assigned to each mission is 1.16, with an average path length of 3.53 steps, which shows an increase of almost 2 steps on the average path length, while at the same time increasing precision.

Second test

Because the enhancement parameters in the last example were very low and did not affect the scores of the categories radically, we tried to increase them slightly, by setting *"Navigation enhancement Value"* as 3 and *"Query enhancement value"* as 7. We kept the score limit as 1.

The results proved what we said about the preceding results given by Crowdflower because the returned precision was 58.2% with an agreement level of 74.4%. This data by themselves don't say much, but once we see that the Skilled evaluators found the precision to be 52.125%, we understand that some of the data returned by Crowdflower have to be wrong. We obviously trust our skilled evaluators more, considering that they spend much more time analysing the mission in order to give the most precise judgements possible. Also, the results given by our skilled evaluators are quite consistent and all of them follow a precise evolution pattern, linked to the changes that we apply to the parameters.

What can be evinced by the data given by our skilled evaluators is that precision increased with the increase of the enhancement values of both queries and navigation data. This is most likely caused by the already underlined fact that the queries, when properly written, should indicate he real information need of the mission. The Navigation data should then be an aid to understand what the query is about.

We also found the average number of categories in each mission Starting category set to be 4.05, almost identical to the preceding test. This is probably due to the score limit not changing in the two tests. Also the average number of Labels for each mission was of 1.15, and the average number of steps to reach the closest goal category for each mission was 3.5.

Third Test

With some in depth analysis of our tool processes, we easily saw that limiting the starting categories in SCS_i to the ones with a score equal to the Highest score amongst all categories, was too restrictive, in particular considering that there might be some other categories with a score very close to the highest score, that should be considered relevant. Therefore we tried to keep the same enhancement values as the Second Test, but changing the score limit to 1.2 times that of the highest score category.

The results given by the Crowdflower evaluators, shows a labelling precision of the 39.99%, with an 82.8% of agreement, while the skilled evaluators reported a precision of 54.25%. Considering that, because of clear inconsistencies, we can't say anything about the changes in precision highlighted by the Crowdflower evaluators, we will limit to analyse the evolution of the precision found by our skilled evaluators.

With an increase of 2.125% over the second test, we can see that the choice of increasing the score limit gave some good results, confirming our hypothesis of some relevant categories not being considered because having a score slightly lower then the highest one.

For this test, the average number of Categories in each mission Starting category set was 5.84, with an increase of 1.79 categories per mission. This is obviously cause by the increase in the score limit, which allows for more categories to be accepted in the Starting Category Set. The average Labels per mission was almost identical to the preceding test with 1.14 labels each. The average number of steps necessary to reach the goal categories from the starting categories was of 3.23.

Fourth Test

In our Fourth test we chose to further increase the enhancement values, setting the Navigation Enhancement value to 25 and the Query enhancement value to 35. Also, we slightly increased the score limit to 1.4 times that of the highest score. This decision was taken following some further testing with the parameters of the preceding test but with an increase score limit. We also tried to increase the score limit even further, but increasing it past 1.4-1.5 resulted in a decrease in precision. The Crowdflower evaluators returned a precision of 63.2%, with an agreement level of 74%. Our skilled evaluators though showed a precision level of 56.125%, increased of 1.875% over the last test. Increasing the score limit and the enhancement values has then allowed us to further increase precision, even though, as pointed out earlier, increasing the score limit further worsens precision, as too many non relevant categories would be taken in consideration.

Also, the Average number of categories in each Starting Category Set was of 6.48. As we have seen in the preceding test, this is due to the score limit increase. At the same time though, the average length of the path to reach the closest goal category for each mission has decreased from 3.23 to 3.06, while the average number of labels for each mission is stable at 1.14 each.

Fifth Test

In our fifth test we chose to increase even more the enhancement values, setting them to 35 and 50 for the Navigation data and the Queries respectively. The score limit was kept at a value of 1.4 times that of the highest value. We chose to increase the enhancement values because it is clear that we couldn't increase the score limit too much, as we would then consider categories that are not relevant enough for our mission (something that was also proved by some further Tests that won't be reported here). At the same time, we don't know exactly how our tool will behave by increasing the enhancement values. Obviously one clear effect of this choice is that the query entities will have very high relevance.

The results showed a precision of 60.8% as result of the Crowdflower evaluation, with an agreement level of 73.6%, while the more reliable skilled evaluators showed a precision of 57.37%. We can then see that the choice of increasing the enhancement values brought an increase in precision, even if limited to 1.25%, of the precision. This proved that the queries represent correctly what the user is looking for most of the times. While at first we thought that a small enhancement value would be enough to obtain optimal results, from this test we can see that, even if not by a great amount, by increasing the enhancement values even more, we increase the precision as well.

The average number of categories in the Starting Category Set of each mission was 6.21, very similar to the last test, which was expected considering that the score limit was unchanged. The average number of steps to reach the closest goal category stayed exactly the same at 3.06 steps, while the average number of labels per mission changed only slightly from 1.14 to 1.15 labels per mission.

Sixth Test

In the sixth test we tried setting the enhancement values very high, in order to understand the level of precision increase that could be obtained this way. We tested the tool by choosing higher values each time, until we found that once we get past a Navigation Enhancement value of 75 and a Query enhancement value of 35, precision almost stops increasing. Also we saw that increasing it too much we have a slight decrease in precision, indicating that we are starting to ignore all those entities that are not present in the query, but of which there are many instances in the Navigation data.

In this case we also chose to give a higher relevance to Navigation data that was also found in the Query, instead of giving an higher enhancement value to the Queries. The results do not change much once we get past enhancement values of 60, or even less, though. In another test we set both enhancement values at 60, and obtained almost the same results. By setting the values so high, we choose to give relevance almost only to entities that are present in the queries, and only consider those present in the Navigation data if the same entity is present in huge quantities.

The results given by the Crowdflower evaluators showed a precision of 70.4%, whit an agreement level of 82.8%. Our skilled evaluators on the other hand have reported a precision level of 61.999%. The increase over the preceding test is of 3.625%, showing therefore an appreciable increase, although obtained by a huge increase in the enhancement values.

At the same time, the number of average categories in the Starting Category Set of each category changed only very slightly from 6.21 to 6.31. The average shortest path length for each mission stayed exactly the same at 3.06 steps, while the average number of labels per mission increased from 1.15 to 1.17.

We can see the various results in Figure 4.4 and 4.5.

Strict Evaluation

During the test evaluation, we also chose to analyse the results of a single evaluator, which proved to be the more consistent in the choices made, and the one that was more strict while evaluating labels. His choices were mostly limited to specific categories that properly represented the mission, while he ignored most general categories that were deemed correct by most of the other skilled evaluators.

The results are shown in Figure 4.6.

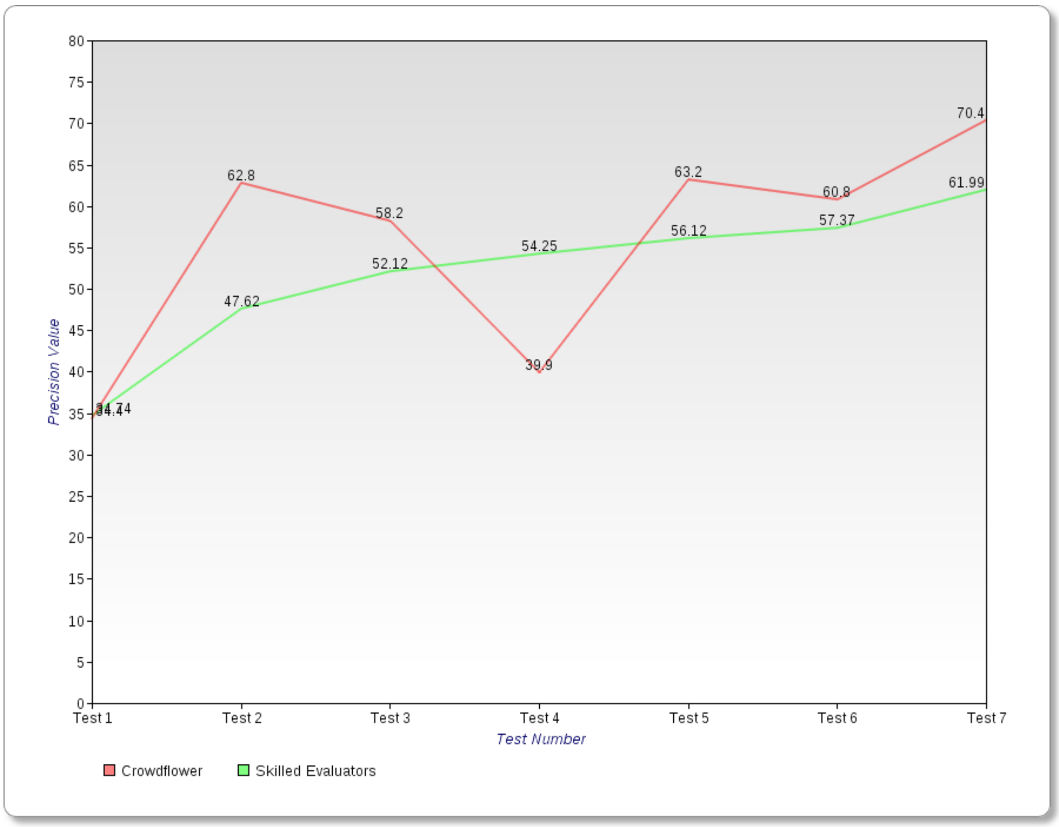


Figure 4.4: The Crowdflower and the Skilled users precision results

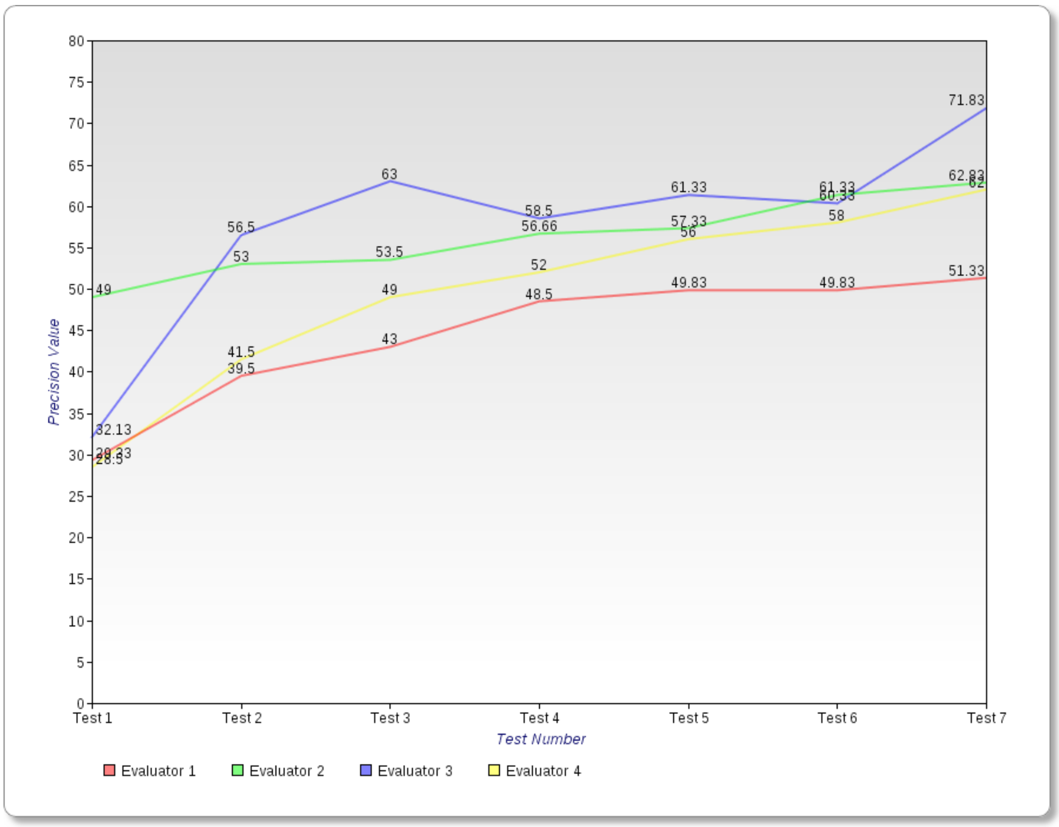


Figure 4.5: The precision results given by the 4 skilled evaluators

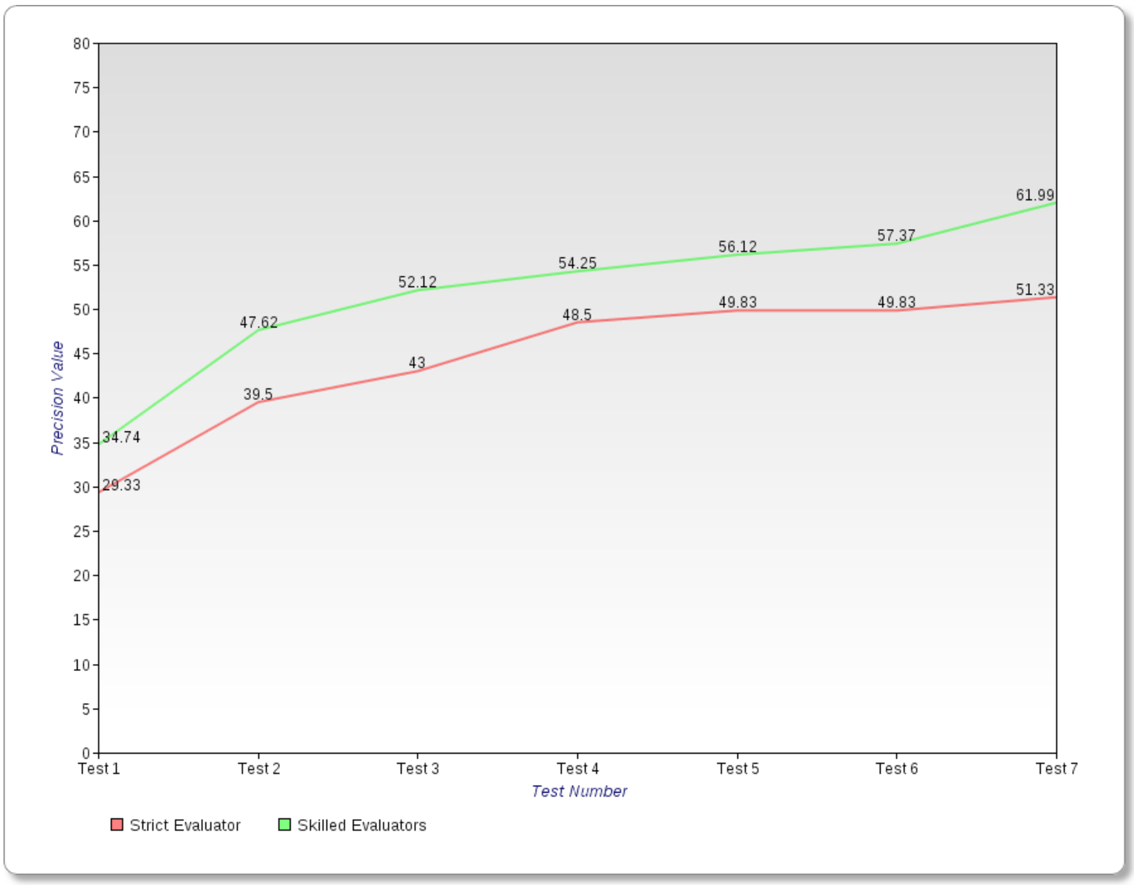


Figure 4.6: The average skilled evaluators' and the strict evaluator's precision results

Analysis of the Tests

Our results highlight a growth in precision which is very pronounced between the baseline and the first 3 test, while the growth decreases in the following Tests, until it almost stops once we reach the last Test.

With a thorough Analysis of the preceding test results we saw that we can obtain a good precision, even though there is a lot of space for improvement. In particular 3 main problems arise from the results.

The given web-pages, which are the home-pages relative to the actual visited web-page, contain mostly non relevant data, because home-pages are normally filled with pictures, flash animations, menus, links, but not an actual extended description of what a website is about. Also, many visited web-pages are Wikipedia web-pages, or yahoo answers pages, or virtual communities, which means that by giving us only the Home-page address, they actually give us information that creates confusion.

Although this point only explains some of the mistakes done by our tool. Other mistakes are due to the structure of the WCG. As a clear example, in our test dataset, there is a mission that have, amongst others similar, the queries ”palate pressure; soft palate pressure; tooth starting to abscess”, obviously talking about health problems related to the mouth. The visited websites were all concerning health and medicine too. Therefore the only category in the Starting Category set was *Palate*, which seems correct. But, the closest Goal category to *Palate* for some reason it's *Art*. *Health*, *Animals*, *Science*, or any other category we might think of that seems more related to *Palate*, is further away. The connection between Palate and art exploits the relation between palate and the human body, and then the representation of the human body in Art. But for a human reading the category *Palate*, the connection is obviously very faint, which is something that it is not possible to understand from the WCG. Some attempts have been made by other authors to define a weight measure, though without appreciable results. This is obviously caused by the fact that the WCG doesn't have predefined levels, it has short-cuts, and it is made by the users, which clearly have multiple opinions on how categories should be linked. This is particularly valid considering that they need to handle a 400.000 categories set.

Another reason for mistakes in the labelling process are caused by wrong entities extracted from the text by Dexter. From time to time, a query is either too short for Dexter to understand the correct context and therefore correctly disambiguate the

Entity. As an example, the mission having as query ”chonda pierce”, and the site ”http://www.chonda.org”, which is about an American Comedian and Singer. Dexter identifies the entity Chonda, a Chinese engine, and therefore it makes it hard to understand that the mission actually is about a Singer. Without many other entities talking about comedy or music though it will be impossible to label the mission correctly.

If we add together the mistakes caused by the WCG structure, by non relevant content in the Web-pages, and by mistakes caused by Dexter, we can explain a vast portion of mistakes, although other mistakes are due to how the queries are written. If a Query is not clear, or if it doesn’t contain the most significant terms for the mission, or also if there are misspells, the tool has a very hard job understanding what the mission is on about. The web pages should have the objective of helping us to understand the actual mission of the queries, but that does not always happen because of the problems we showed before.

The obtained results are interesting nonetheless, and they show in particular how much precision has risen from the baseline approach, but also from the basic approach shown in the first test up to the optimal values we found.

Chapter 5

Conclusions and Future Work

In this thesis, we presented a new approach for mission labelling, by exploiting the Wikipedia Category Graph. After adapting the dataset, extracting the necessary data and developing the tool, we tested it with a very large mission log, developed by Hagen et al. at the university of Weimar. For the evaluation, we relied on two distinct groups: a skilled set of evaluators, and the Crowdflower, a crowdsourcing company offering human workforce to perform repetitive tasks that can't be done by a machine.

To develop the algorithm we analysed each step necessary to get from the extracted entities, to the goal categories, and tried to exploit co-occurrence, multiple instances, and score of both entities and categories.

Three main problems arose from the analysis of the results: web pages in the mission logs should be the actual visited pages. Also, they should be built following a precise structure, so that the meaningful parts of the web-page are held in a single HTML tag, in alternative a much easier possibility is that of starting to actually use the meta tags in HTML to indicate keywords. The second main problem is the sometime chaotic WCG structure. While asking the administrators of Wikipedia to change the structure is not feasible, we can try to develop a weight determining how much a category is related to its parent category. The final problem is that, from time to time, Dexter, or any entity linking software, will link a spot to the wrong entity, because of the lack of context. This matter needs further researches to understand how it should be handled.

We believe that this work will be of interested for anyone handling Mission labelling, but also query labelling, because of the analysis of the problems related to the Wikipedia Category Graph, which is being widely used for categorization nowadays.

Also the use of Dexter in our project showcased the potential of this and of similar application, that might find extensive use in future labelling tools.

Future works include the development of a weight measure for WCG edges, giving us a chance to differentiate between relevant and less relevant links between two categories. Alternatives to the formulas we employed in our tool can be developed too and other parameters can be tested, in order to try to obtain a better precision. A possible idea is also that of employing a tool for word stemming, or to auto correct words in the queries that do not return any result when searched for alone. This way we could avoid problems caused by search misspells. Also, a new mission log, or an adaptation of the one we used, should be developed with the actual clicked webpages, in order to better evaluate the precision of similar tools. A ground truth for the evaluation of future tools using this Mission log should also be defined, otherwise every evaluation will be given based on subjective opinions.

Acknowledgements

I would like to thank my supervisor Professor Salvatore Orlando and my co-supervisor Dr. Gabriele Tolomei for helping me with this thesis, providing me with the idea and assisting me in its development up to the final result. Also, i want to thank Dr. Diego Ceccarelli of the university of Pisa for providing me with the Dexter tool, and for continuously giving me technical assistance.

I also would like to thank all my family for supporting me in both a psychological and practical way through all my academic career and for having always believed in me. For my father and my mother who took care of me and travelled a long way to be here in this important day, and to my grandparents who were at my side through all of this journey.

For the technical advices given, i would like to thank my friends Alessandro Tolomio and Alessandro Savoca, who were always available every time i needed technical support during the development of the software.

A special thanks though goes to all my friends that have always given me strength, and that, whether they were far or near, never ceased to help me and to cheer me up when my spirit was down.

Bibliography

[1] http://lucene.apache.org/core/.

[2] http://wordnet.princeton.edu.

[3] http://www.dmoz.com.

[4] http://www.lemurproject.org.

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

[6] M. Alemzadeh, R. Khoury e F. Karray. Exploring Wikipedia's Category Graph for Query Classification. In M. Kamel, F. Karray, W. Gueaieb e A. M. Khamis, cur., *AIS*, vol. 6752 di *Lecture Notes in Computer Science*, pp. 222–230. Springer, 2011. ISBN 978-3-642-21537-7.

[7] M. AlemZadeh, R. Khoury e F. Karray. Query Classification using Wikipedia's Category Graph. *Journal Of Emerging Technologies In Web Intelligence*, vol. 4(3), August 2012.

[8] A. Bagga e B. Baldwin. Entity-Based Cross-Document Coreferencing Using the Vector Space Model. In *Proceedings of the 17th International Conference*

on Computational Linguistics, pp. 79–85. Association for Computational Linguistics, Morristown, NJ, USA, 1998.

[9] S. M. Beitzel, E. C. Jensen, A. Chowdhury, D. A. Grossman e O. Frieder. Hourly analysis of a very large topically categorized web query log. In M. Sanderson, K. Järvelin, J. Allan e P. Bruza, cur., *SIGIR*, pp. 321–328. ACM, 2004. ISBN 1-58113-881-4.

[10] S. M. Beitzel, E. C. Jensen, D. D. Lewis, A. Chowdhury e O. Frieder. Automatic Classification of Web Queries Using Very Large Unlabeled Query Logs. *ACM Trans. Inf. Syst.*, vol. 25(2), apr. 2007.

[11] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis e S. Vigna. The query-flow graph: model and applications. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge mining*, pp. 609–618. ACM, New York, NY, USA, 2008. ISBN 978-1-59593-991-3.

[12] A. Broder. A taxonomy of Web search. *ACM SIGIR Forum*, vol. 36(2):pp. 3–10, 2002.

[13] R. Bunescu e M. Pasca. Using Encyclopedic Knowledge for Named Entity Disambiguation. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 9–16. Trento, Italy, April 2006.

[14] D. Carmel, H. Roitman e N. Zwerdling. Enhancing Cluster Labeling Using Wikipedia. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pp. 139–146. ACM, New York, NY, USA, 2009. ISBN 978-1-60558-483-6.

[15] D. Ceccarelli, C. Lucchese, S. Orlando, R. Perego e S. Trani. Dexter: An open source framework for entity linking. In *Proceedings of the Sixth International*

Workshop on Exploiting Semantic Annotations in Information Retrieval, ESAIR '13, pp. 17–20. ACM, New York, NY, USA, 2013. ISBN 978-1-4503-2413-7.

[16] S. Cucerzan. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of EMNLP-CoNLL*, vol. 2007, pp. 708–716. 2007.

[17] D. Cutting, D. R. Karger, J. O. Pedersen e J. W. Tukey. Scatter/Gather: A Cluster-Based Approach to Browsing Large Document Collections. In *SIGIR-92*, pp. 318–329. ACM, Copenhagen, Denmark, 1992.

[18] D. Donato, F. Bonchi, T. Chi e Y. Maarek. Do you want to take notes?: identifying research missions in Yahoo! search pad. In *Proceedings of the 19th international conference on World wide web*, pp. 321–330. ACM, New York, NY, USA, 2010. ISBN 978-1-60558-799-8.

[19] M. Dredze, P. McNamee, D. Rao, A. Gerber e T. Finin. Entity Disambiguation for Knowledge Base Population. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pp. 277–285. Association for Computational Linguistics, Stroudsburg, PA, USA, 2010.

[20] J. Farina. *Assegnamento Automatico Di Macrocategorie Agli Articoli Di Wikipedia*. Bachelor thesis, Politecnico Di Milano, 2009-2010.

[21] P. Ferragina e U. Scaiella. TAGME: on-the-fly annotation of short text fragments (by Wikipedia entities). *CoRR*, vol. abs/1006.3498, 2010.

[22] E. Gabrilovich e S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pp. 1606–1611. 2007.

[23] E. Gabrilovich e S. Markovitch. Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis. *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pp. 6–12, 2007.

[24] D. Gayo-Avello. A survey on session detection methods in query logs and a proposal for future evaluation. *Inf. Sci.*, vol. 179(12):pp. 1822–1843, 2009.

[25] F. Geraci, M. Pellegrini, M. Maggini e F. Sebastiani. Cluster Generation and Cluster Labelling for Web Snippets: A Fast and Accurate Hierarchical Solution. In *Proceedings of the 13th International Conference on String Processing and Information Retrieval*, SPIRE'06, pp. 25–36. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 3-540-45774-7, 978-3-540-45774-9.

[26] L. Gravano, V. Hatzivassiloglou e R. Lichtenstein. Categorizing Web Queries According to Geographical Locality. In *12th ACM Conference on Information and Knowledge Management (CIKM 2003)*, pp. 325–333. ACM Press, New York, NY, USA, November 3-8 2003.

[27] J. Guo, X. Cheng, G. Xu e X. Zhu. Intent-aware query similarity. In C. Macdonald, I. Ounis e I. Ruthven, cur., *CIKM*, pp. 259–268. ACM, 2011. ISBN 978-1-4503-0717-8.

[28] M. Hagen, J. Gomoll, A. Beyer e B. Stein. From Search Session Detection to Search Mission Detection. In *Proceedings of the 10th Conference on Open Research Areas in Information Retrieval*, OAIR '13, pp. 85–92. Le Centre De Hautes Etudes Internationales d'Informatique Documentaire, Paris, France, France, 2013. ISBN 978-2-905450-09-8.

[29] M. Hagen, B. Stein e T. Rüb. Query Session Detection As a Cascade. In *Proceedings of the 20th ACM International Conference on Information and Knowl-*

edge Management, CIKM '11, pp. 147–152. ACM, New York, NY, USA, 2011. ISBN 978-1-4503-0717-8.

[30] C. Harrison. Mathematics clusterball. http://www.chrisharrison.net/index.php.

[31] D. He e A. Göker. Detecting Session Boundaries from Web User Logs. In *Proceedings of the BCS-IRSG 22nd annual colloquium on information retrieval research*, pp. 57–66. Cambridge, UK, 2000.

[32] J. Hu, G. Wang, F. Lochovsky, J. tao Sun e Z. Chen. Understanding user's query intent with wikipedia. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pp. 471–480. ACM, New York, NY, USA, 2009. ISBN 978-1-60558-487-4.

[33] B. Jansen e A. Spink. How are we searching the World Wide Web? A comparison of nine search engine transaction logs. *Information Processing and Management*, vol. 42(1):pp. 248–263, 2006.

[34] T. Joachims. Making Large-Scale SVM Learning Practical. In B. Schölkopf, C. J. Burges e A. Smola, cur., *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, MA, USA, 1999.

[35] R. Jones e K. L. Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pp. 699–708. ACM, New York, NY, USA, 2008. ISBN 978-1-59593-991-3.

[36] I.-H. Kang e G. Kim. Query type classification for web document retrieval. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pp. 64–71. ACM, New York, NY, USA, 2003. ISBN 1-58113-646-3.

[37] Z. T. Kardkovács, D. Tikk e Z. Bánsághi. The Ferrety Algorithm for the KDD Cup 2005 Problem. *SIGKDD Explor. Newsl.*, vol. 7(2):pp. 111–116, dic. 2005.

[38] R. Khoury. Query Classification Using Wikipedia. *Int. J. Intell. Inf. Database Syst.*, vol. 5(2):pp. 143–163, mar. 2011.

[39] A. Kotov, P. N. Bennett, R. W. White, S. T. Dumais e J. Teevan. Modeling and analysis of cross-session search tasks. In W.-Y. Ma, J.-Y. Nie, R. A. Baeza-Yates, T.-S. Chua e W. B. Croft, cur., *SIGIR*, pp. 5–14. ACM, 2011. ISBN 978-1-4503-0757-4.

[40] S. Kulkarni, A. Singh, G. Ramakrishnan e S. Chakrabarti. Collective Annotation of Wikipedia Entities in Web Text. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pp. 457–466. ACM, New York, NY, USA, 2009. ISBN 978-1-60558-495-9.

[41] S. Kullback e R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, vol. 22(1):pp. 79–86, 1951.

[42] T. K. Landauer, P. W. Foltz e D. Laham. An introduction to latent semantic analysis. *Discourse Processes*, vol. 25:pp. 259–284, 1998.

[43] C. Leacock e M. Chodrow. Combining local context and WordNet similarity for word sense identification. In C. Fellbaum, cur., *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[44] C. Lucchese, S. Orlando, R. Perego, F. Silvestri e G. Tolomei. Identifying task-based sessions in search engine query logs. In I. King, W. Nejdl e H. Li, cur., *WSDM*, pp. 277–286. ACM, 2011. ISBN 978-1-4503-0493-1.

[45] C. Lucchese, S. Orlando, R. Perego, F. Silvestri e G. Tolomei. Discovering tasks from search engine query logs. *ACM Trans. Inf. Syst.*, vol. 31(3):pp. 14, 2013.

[46] C. Lucchese, S. Orlando, R. Perego, F. Silvestri e G. Tolomei. Modeling and Predicting the Task-by-task Behavior of Search Engine Users. In *Proceedings of the 10th Conference on Open Research Areas in Information Retrieval*, OAIR '13, pp. 77–84. Le Centre De Hautes Etudes Internationales d'Informatique Documentaire, Paris, France, France, 2013. ISBN 978-2-905450-09-8.

[47] G. S. Mann e D. Yarowsky. Unsupervised Personal Name Disambiguation. In W. Daelemans e M. Osborne, cur., *Proceedings of the* 7^{th} *Conference on Natural Language Learning (CoNLL-2003)*, pp. 33–40. Edmonton, Canada, mag. 2003.

[48] O. Medelyan, I. H. Witten e D. Milne. Topic indexing with Wikipedia. In *Proceedings of the Wikipedia and AI workshop at AAAI-08*. AAAI, 2008.

[49] R. Mihalcea e A. Csomai. Wikify!: Linking Documents to Encyclopedic Knowledge. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pp. 233–242. ACM, New York, NY, USA, 2007. ISBN 978-1-59593-803-9.

[50] D. Milne e I. H. Witten. Learning to Link with Wikipedia. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge mining*, pp. 509–518. ACM, New York, NY, USA, 2008. ISBN 978-1-59593-991-3.

[51] H. C. Ozmutlu e F. Çavdur. Application of automatic topic identification on excite web search engine data logs. *Inf. Process. Manage.*, vol. 41(5):pp. 1243–1262, 2005.

[52] G. Pass, A. Chowdhury e C. Torgeson. A picture of search. In *Proceedings of the 1st International Conference on Scalable Information Systems*, InfoScale '06. ACM, New York, NY, USA, 2006. ISBN 1-59593-428-6.

[53] S. P. Ponzetto e M. Strube. Deriving a large scale taxonomy from Wikipedia. In *Proceedings of the 22nd national conference on Artificial intelligence - Volume 2*, AAAI'07, pp. 1440–1445. AAAI Press, 2007. ISBN 978-1-57735-323-2.

[54] D. R. Radev, H. Jing e M. Budzikowska. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. *NAACL/ANLP Workshop on Automatic Summarization, Seattle, WA, April 30, 2000*, mag. 2000.

[55] F. Radlinski e T. Joachims. Query Chains: Learning to Rank from Implicit Feedback. In *ACM SIGKDD International Conference On Knowledge Discovery and Data Mining (KDD)*, pp. 239–248. 2005.

[56] H. Raghavan, J. Allan e A. McCallum. An exploration of entity models, collective classification and relation description. *KDD Workshop on Link Analysis and Group Detection*, 2004.

[57] Y. Ravin e Z. Kazi. Is Hillary Rodham Clinton the President? Disambiguating Names across Documents. In *Proceedings of the ACL 1999 Workshop on Coreference and its Applications*. giu. 1999.

[58] D. Shen, R. Pan, J.-T. Sun, J. J. Pan, K. Wu, J. Yin e Q. Yang. Q2C@UST: Our Winning Solution to Query Classification in KDDCUP 2005. *SIGKDD Explor. Newsl.*, vol. 7(2):pp. 100–110, dic. 2005.

[59] W. Shen, J. Wang, P. Luo e M. Wang. LINDEN: Linking Named Entities with Knowledge Base via Semantic Knowledge. In *Proceedings of the 21st Interna-*

tional Conference on World Wide Web, WWW '12, pp. 449–458. ACM, New York, NY, USA, 2012. ISBN 978-1-4503-1229-5.

[60] X. Shen, B. Tan e C. Zhai. Implicit user modeling for personalized search. In *CIKM*, pp. 824–831. 2005.

[61] C. Silverstein, H. Marais, M. Henzinger e M. Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, vol. 33(1):pp. 6–12, 1999.

[62] F. Silvestri. Mining Query Logs: Turning Search Usage Data into Knowledge. *Foundations and Trends in Information Retrieval*, vol. 4(1-2):pp. 1–174, 2010.

[63] A. Spink, M. Park, B. J. Jansen e J. Pedersen. Multitasking during web search sessions. *Inf. Process. Manage.*, vol. 42(1):pp. 264–275, 2006.

[64] D. Vogel, S. Bickel, P. Haider, R. Schimpfky, P. Siemen, S. Bridges e T. Scheffer. Classifying Search Engine Queries Using the Web As Background Knowledge. *SIGKDD Explor. Newsl.*, vol. 7(2):pp. 117–122, dic. 2005.

[65] N. Wacholder, Y. Ravin e M. Choi. Disambiguation of proper names in text. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, ANLC '97, pp. 202–208. Association for Computational Linguistics, Stroudsburg, PA, USA, 1997.

[66] J.-Y. J.-R. Wen e H.-J. Zhang. Query Clustering Using User Logs. *ACM Transactions on Information Systems (ACM TOIS)*, vol. 20(1):pp. 59–81, 2002.

[67] I. H. Witten e D. Milne. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In *Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy, AAAI Press, Chicago, USA*, pp. 25–30. 2008.

[68] T. Zesch, C. Müller e I. Gurevych. Extracting lexical semantic knowledge from wikipedia and wiktionary. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*. Marrakech, Morocco, mag. 2008. electronic proceedings.