Visualization of Thesaurus-Based Web Search

186.834 Praktikum aus Visual Computing
SS 2017

Michael Mazurek

27th October 2017

1 Abstract

The general functions of current web search engines are well established. A box is provided in which to type the queries and the engine returns a result list which users can evaluate. The autocomplete suggestions assist users in defining their problems, however there is a lack of support for an iterative manual refinement of the query. This additional aid can be beneficial when users not know the exact terms to describe the concept they are looking for. Therefore, the goal of this project is to show searchers how a slight variation of the query changes the results. With this information, they then can perform a targeted refinement of the query to access useful information sources. To achieve this goal, each part of the searcher's query is varied with a thesaurus that provides synonyms for the individual query terms. While performing the user's original query in a normal fashion, variations of this query are conducted in the background. To provide a concise visual summary of the query results, text mining techniques are performed on all gathered results to retrieve the most important key terms for each query variation. This procedure results in a visual overview of what the searcher's query finds together with what could be found with a slight variation of the query. Additional queries should make users aware that alternative queries may be more appropriate when their original query is poorly formulated. In conjunction with some interaction tools, the goal is to reduce the burden of refining search queries and therefore making searching the web less complex.

2 Introduction

One of the fundamental problems of information retrieval is word mismatch, a problem coined by Furnas et al. [8]. Simply put, it means that the users formulation of the searched topic does not match with the authors anticipated concepts. Further refining the query often tends to be a tedious task, as current search engines provide little guidance on how to change the query to get to the desired output. This process has to be done manually by assessing each query refinement with respect to the retrieved results.

To overcome the above mentioned difficulty, a significant number of researchers have studied ways for assisting users in the query selection process. This task is called query expansion (or query refinement), as it attempts to refine the user-issued queries by expanding them with semantically similar terms. Most of these approaches rely on the use of thesauri or other types of semantic resources from which they obtain alternative wording for refining queries. These alternative query formulations are often provided to the users in the form of a list. They then can decide to append or substitute with suggested terms or ignore the suggestions altogether.

The goal of this project is to support the user to find fitting alternative query formulations when word mismatches happen. However, beyond providing formulations, the result of this project should also convey information about how the generated queries relate to the user-issued query. Users should be able to realize the adequacy of their self-selected query terms, and on the other side, see if the system selected terms are able to catch their search intentions more precisely. To solve this task, an extension to the web browser Chrome was developed that automatically expands user-issued queries and visualizes the results of all queries in an Untangled Euler Diagram [14]. By adding a visualization to a common search engine, the users should get the insight into the query terms and be able to quickly single out the most promising query.

3 Related Work

As the project goals suggest, this project revolves around two fields of study. The first one is query expansion, a subfield of Information Retrieval (IR), as the Chrome extension is providing alternative queries for the user. The latter is the visualization, which displays the results of the queries alongside with the queries to convey the correlations between them. In the following section related work from both fields will be discussed as well as how our approach is different.

Query expansion is a research topic that is researched since the late sixties. The approaches of query expansion can be categorized into two areas, depending on which information the expansion is based on.

One possible source of information is the corpus that the user searches his information in. These approaches are called *query expansions with corpus dependent knowledge models.* The concept of relevance feedback is a fairly established technique in this area as outlined by Salton and Mcgill [15]. Users provide relevance information by tagging documents of an initial query. Subsequently, these documents are used to mine for relevant terms. Further research concentrated on more automated approaches by obtaining context from the document collection using stemming, clustering and co-occurrence. A technique to determine the relevancy of a specific term by its co-occurrence with a general concept term was proposed by Chu et al. [5]. In their approach, general concept terms are substituted from the query. The substitution terms are concept terms used in the corpus that co-occur with the key query concept. Besides using co-occurrence also lexical networks can be used as sources for deriving context. Sanderson and Lawrie [16] present an approach which models a concept hierarchy of the corpus. A network of a concept hierarchy is automatically generated from the document collection by extracting salient words from top ranked documents. By using a subsumption function the hierarchy between these concepts is determined.

In corpus dependent techniques, the content of the corpus is analysed to extract canidate terms, which can only work if there are sufficient relevant documents to work with. Therefore *query expansions with corpus independent knowledge models* are the second area of research. Voorhees [17] used WordNet [7] to conduct experiments on small single domain TREC (Text REtrieval Conference) collections, which are datasets to conduct tests on retrieval performance of information retrieval systems. In this technique, the corpus is annotated with synonym sets from an ontology that are then used to expand the query. Beyond synonyms and hypernyms, also other relations can be used for query expansion. Navigle and Velardi [13] suggest the use of relations in concept definitions. In their work, an ontology is used to extract the semantic domain of a word and then the query is expanded further by co-occurring words.

To address the information retrieval problem further, a visualization is added to the extension, which indicates the relations between the created queries. In particular two areas of visualization techniques are of interest: visualizations of query expansions and visualization of query results.

Visualizing relations between queries is often done through variations of node-link diagrams, as these relations are based on ontology knowledge graphs or binary co-occurrence relations that can be mapped easily onto links. One such an approach for visualizing query extensions is the work of Hoeber et al. [11]. They encode three classes of nodes (concepts, unselected terms, selected terms) with color and link terms to concepts based on the degree to which the terms represent the intension of the concept. Users can then interact with the graph to modify the query, by adding or removing terms and concepts. The graph encoding of Kozanidis et al. [12] is another example. In their graph encoding, nodes represent queries. The links between them depict the semantic distance between the nodes based on the WordNet [7]. Clicking on connected nodes results in a Boolean "and" combination of the queries, disconnected nodes are combined with an Boolean "or".

Also the visualization of query results is an established research topic, in particular visualizations that provide a visual representation for a whole document or document surrogate are of interest. Gomez et al. [9] proposes a snippet-based visualization for query results based on high-dimensional term frequency vectors that are mapped to the visual space. The snippets are placed into the visual space based on the projection, and the positions are then optimized to avoid overlap and unused space. A different approach is explored by Hearst [10]. The contents of the document are divided into fixed blocks and the frequency of the query terms is color coded into each block. This results in a set of bars (one for each document) with widths relative to the length of the documents, and heights relative to the number of query terms.

The apporaches summarized here touch upon aspects related to our work. In the presented terminology our query expansion is a simple corpus independent approach that relies on a thesaurus and permutation of synonyms to create suggested queries. What makes our work different from existing techniques is the visualization that encodes the queries *and* the results. It is a visualization of query expansion results that assists the user to realize the underlying correlation between the self selected query and suggested queries by the system. As presented in this section, previous approaches covered only the relation of multiple queries *or* the relation of query results to each other. Our approach combines these two methods to apply a visualization that can encode how the variation of the query leads to a variation of the result instead of only encoding the variation of the query, or the summary of the results.

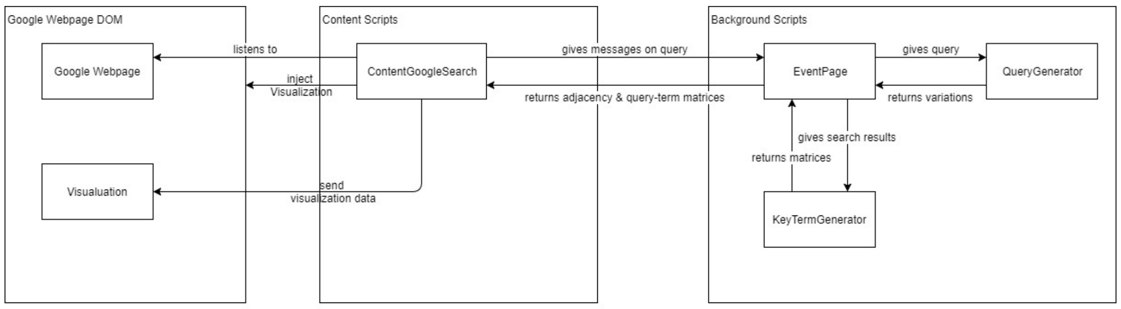

Figure 1: The components of the Extension and how they are connected.

4 Implementation

This Chrome extension was implemented in HTML and JavaScript using a multitude of libraries. Generally, the tool consists of a content script and an event page. How the components are connected can be seen in Figure 1. Reading out the query from the Google Search Page and the injection the visualization is done by a content script. The persistent event page generates additional queries and mines key terms for the visualization. The following sections will cover the query expansion, key term generation and the visualization in detail.

4.1 Query Expansion

Generating additional queries from the original queries works by following some basic steps. First, the query is split into unigrams and part-of-speech tags with the help of the compromise.js library [4] to determine which words to vary. All nouns, verbs and adjectives are searched for in a thesaurus provided by the datamuse library [3], the synonyms are then used to generate query permutations. Permuting new queries is a random process, the algorithm selects for each word position in the query a synonym or keeps the original word. As this process can produce duplicate queries it repeats until the desired number of queries is reached or the number of iterations reaches two times the number of possible query permutations.

4.2 Key Term Generation

After the Query Expansion step the queries are sent to the Google Custom Search API [2] to retrieve the titles and snippets found by the search machine. Each query provides ten titles and snippets at a time which are then sent to the key term generator to generate the data needed for the visualization. The procedure starts with stemming and stop-word removal of the whole corpus followed by the generation of dictionaries for all queries to calculate the document frequency (i.e: the total number of queries containing a specific term). In the next step, these dictionaries are used to generate the term frequencies as well as the tfidf. For the generation of the visualization data, the words of the corpus are ordered by one of these scores. The first n terms of the list are then used to generate nodes that are scaled based on their scores. For all these words co-occurrences are computed. Words get linked when they are found in the same query. Based of the number of co-occurrence in queries, the link between them gets stronger. Together with the described information, it is also sent in which query the terms are present.

4.3 Visualization

With the data structure generated in the previous step, the Untangling Euler Diagram from Riche et al. [14] can now be initialized and drawn. The initialization step starts with an algorithm that splits the queries into multiple parts so that there are no overlaps of non-intersecting queries. After that, enclosures of the separated parts are joined together using links. A simple technique is used to draw a single filled path centered on the barycenter of separated enclosures. To draw the path, a support structure is calculated. It consist of support lines from the barycenter to the center of each enclosure and lines perpendicular to these inside the enclosure. Splines are then computed using these support lines. The splitting algorithm, as well as the link representation algorithm, is explained in more detail in Riche's et al.'s paper.

Lastly, the cola.js library [1] is used to generate the layout of the visualization using IPSep-CoLa [6], a force-directed layout algorithm with separation constraints. The selected terms from the key term generation step, together with their co-occurrences, define the input nodes and edges for the force algorithm. Additionally, a grouping of input nodes is specified that is computed with the algorithm proposed by Riche. To label the groups with the query, additional input nodes are added to the computation and placed in the layout by inequality constraints. Inequality constraints ensure that there is a specified gap between the center points of two nodes on a chosen axis. The label input nodes are placed at the top by formulating such separation constraints that force all other nodes in the group beneath the label.

5 User Guide

This last section revolves around the set up of the chrome extension and the general use case of the tool. First, a guide is given on how to set up a chrome extension. Subsequently, an introduction follows on how to set up the options page and what all of the settings mean. Lastly, a use case is presented to show how the work flow of the application looks and what benefits it adds to the querying process.

5.1 Chrome extension setup and initializing the options page

To add an custom extension to the Chrome browser, the user has first to navigate to the extensions menu. The menu can be found by clicking the top right box in the menu bar and selecting *more tools/extensions*. In this menu the *developer mode* has to be activated, on the top right a check box can be found to initialize the mode. When the check box is clicked, a set of buttons appear. The *load unpacked extension* button can then be used to load an extension by selecting the appropriate directory. When it is loaded, the extension must be initialized. This can be done by accessing the options page of the extension.

On the options page, the user can initialize the visualization with custom options or press the *reset* button to get default values. As the debug settings are more complex, additional insight into these extended settings is given here. The *debug* check box enables all debug options beneath it. When it is unchecked the extension operates as described in Section 4. The Google API is queried in the query expansion process and the visualization uses the data that is generated by the key term generator. In this mode, all queries are cashed in chrome storage for later use in debug mode. Checking the *debug* check box will cause the application to skip the Google API. Instead of generating dynamic data, debug data is loaded which can be selected in the *select debug query* drop-down menu. Furthermore, the user can also decide to use manually created test data. To access this data the *change data* switch field has to be changed to *test data*.

5.2 Example Use Case

In this last section we examine a use case to show how the work flow of the application looks and what benefits it adds to the querying process. This use case revolves around the search term *information vis*, as the user is interested in information around the concept of information visualization. To research this term, the user enters it into the Google Search Page. The work flow of querying the search page stays the same. However the extension provides additional information by displaying a visualization next to the result list.

As can be seen in the Figure 2, the visualization shows the original query in blue and generated ones are placed around it. Looking at the terms that are only in the blue enclosure conveys concepts that are only present in the original query. Examining further reveals that some of these terms, like *vaccine*, *visa* and *viss*, relate to other concepts than the searched concept of information visualization. Therefore, we should decide for an alternative query, that encloses all terms from the original query that interest us without introducing terms that do not fit the searched concept. Such an example is the *info vis* query, as *JavaScript*, *infovis* and *paper* fit the concept. As seen in this use case, it is possible to derive concepts from the key terms and see the relations between the existing concepts through the query enclosures.

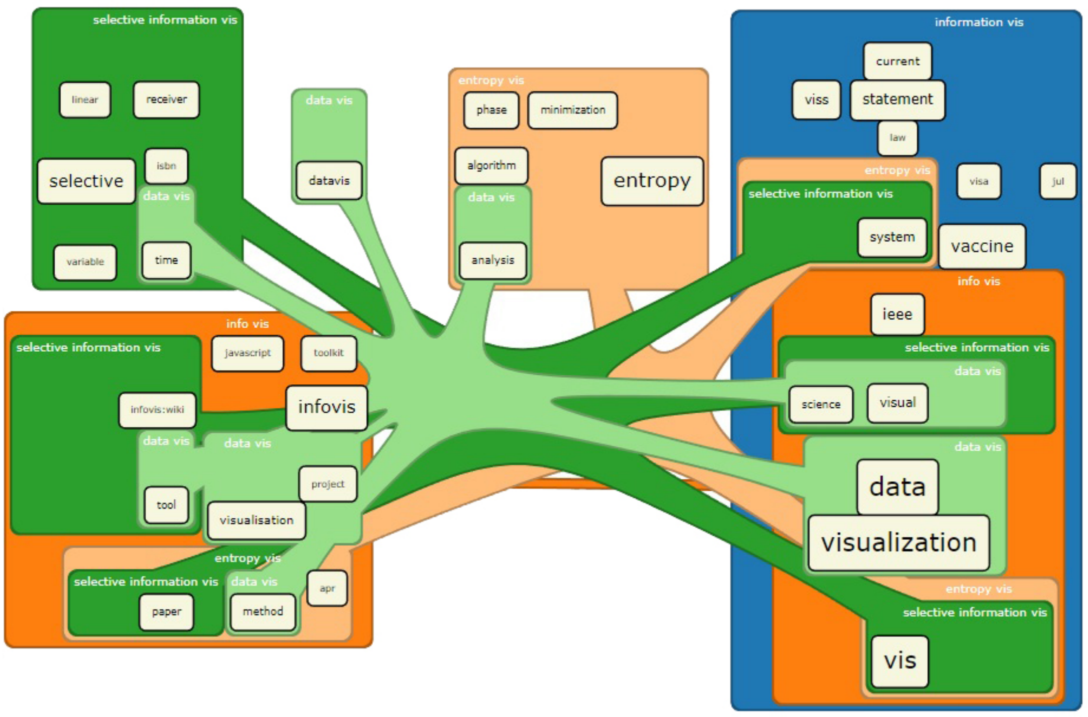

Figure 2: The generated visualization for the use case.

References

[1] Constraint-based layout in the browser. `http://marvl.infotech.monash.edu/webcola/`. Accessed: 2017-09-21.

[2] Custom search json/atom api. `https://developers.google.com/custom-search/json-api/v1/overview`. Accessed: 2017-09-21.

[3] Datamuse api. `https://www.datamuse.com/api/`. Accessed: 2017-09-21.

[4] Natural language processing in javascript. `https://github.com/nlp-compromise/compromise`. Accessed: 2017-09-21.

[5] Wesley W Chu, Zhenyu Liu, and Wenlei Mao. Textual document indexing and retrieval via knowledge sources and data mining. *Communication of the Institute of Information and Computing Machinery (CIICM), Taiwan*, 5(2), 2002.

[6] Tim Dwyer, Yehuda Koren, and Kim Marriott. Ipsep-cola: An incremental procedure for separation constraint layout of graphs. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):821–828, 2006.

[7] Christiane Fellbaum. *WordNet.* Wiley Online Library, 1998.

[8] George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11):964–971, 1987.

[9] Erick Gomez-Nieto, Frizzi San Roman, Paulo Pagliosa, Wallace Casaca, Elias S Helou, Maria Cristina F de Oliveira, and Luis Gustavo Nonato. Similarity preserving snippet-based visualization of web search results. *IEEE transactions on visualization and computer graphics*, 20(3):457–470, 2014.

[10] Marti A Hearst. Tilebars: visualization of term distribution information in full text information access. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 59–66. ACM Press/Addison-Wesley Publishing Co., 1995.

[11] Orland Hoeber, Xue-Dong Yang, and Yiyu Yao. Visualization support for interactive query refinement. In *Web Intelligence, 2005. Proceedings. The 2005 IEEE/WIC/ACM International Conference on*, pages 657–665. IEEE, 2005.

[12] Lefteris Kozanidis, Paraskevi Tzekou, Nikos Zotos, Sofia Stamou, and Dimitris Christodoulakis. Ontology-based adaptive query refinement. In *WEBIST (2)*, pages 43–50, 2007.

[13] Roberto Navigli and Paola Velardi. An analysis of ontology-based query expansion strategies. In *Proceedings of the 14th European Conference on Machine Learning, Workshop on Adaptive Text Extraction and Mining, Cavtat-Dubrovnik, Croatia*, pages 42–49, 2003.

[14] Nathalie Henry Riche and Tim Dwyer. Untangling euler diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1090–1099, 2010.

[15] Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986.

[16] Mark Sanderson and Dawn Lawrie. Building, testing, and applying concept hierarchies. *Advances in information retrieval*, pages 235–266, 2002.

[17] Ellen M Voorhees. Query expansion using lexical-semantic relations. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 61–69. Springer-Verlag New York, Inc., 1994.