Concept Tagging and Voting for an Improved Search Experience

David Stevens
GATech OMSCS Student
225 North Ave NW
Atlanta, GA 30332
+1-706-955-5952
Stevens.David.Allen@Gmail.com

Ali Murtaza Shaikh
GATech OMSCS Student
225 North Ave NW
Atlanta, GA 30332
+1-713-373-7161
ali.mur@gatech.edu

Aaron Buxbaum
GATech OMSCS Student
239 W 26th Street, Apt 3W
New York, NY 10001
+1-518-289-0123
me@aaronbuxbaum.com

ABSTRACT

We present a concept for search engines which harnesses the powers of artificial intelligence.For example, we present the use-case of a search engine for finding research papers in any academic field. Our search engine utilizes IBM Watson to extract concepts from research papers; it uses a feedback mechanism (voting) to improve search results over time.

CCS CONCEPTS

• Computing methodologies~Machine learning algorithms • ***Human-centered computing~Natural language interfaces*** • ***Information systems~Relational database model***

KEYWORDS

IBM Watson; Concept Extraction; Search Engine; Search Query; Algorithm; Research Paper; Database; mySQL; AI; Natural Language; Knowledge Base; Relational Database.

1. INTRODUCTION

Finding and reading relevant research papers is an integral part of academic research. Scholars use various methods to find relevant research papers, but by far most common is a text-based search engine. Unfortunately, that means that the user must know exactly which words exist in the paper in order to find it. Often, papers are not tagged well, or do not contain the precise words that the user requests; this makes the process of finding relevant papers as a researcher a significant challenge. In addition, the current products have no mode of improvement over time; they do not utilize user feedback to increase a user's ability to find relevant papers. Search systems are often specialized for just one academic area (for example, biomedical sciences), which eliminates the possibility of discovering information that may be classified under a different field, even if they are related conceptually. This problem compounds upon itself depending on how granular the journals are, since the journal may be too "niche" to include information that is, in practice, very closely related to the desired subject information.

2. EXISTING SOLUTIONS

There are several good academic search solutions publicly available on the Internet. In the following section, we will briefly discuss our favorites ones, and how they approach the problem of searching for information within academia.

1. Bielefield Academic Search Engine: Bielefield Academic Search Engine allows users to search over 100 million research documents, to favorite specific documents, and to access search history. It also includes filters, which can refine search results by author, subject, etc [1].

2. **Microsoft Academic:** Microsoft Academic is a semantic inference-based search engine. It utilizes graphing algorithms in order to increase search speed, as well as Microsoft Bing's web crawlers to crawl the web and index new research papers [2].

3. Google Scholar: Google Scholar is a search engine for approximately 160 million scholarly documents [3]. Since its ranking algorithm puts a high weight on citations, a highly cited paper might show up higher in the results, even when it is less relevant than other papers [10].

4. EBSCO: EBSCO is a search engine that provides a fee-based full text academic search engine [4].

5. Mendeley: Unlike the others, Mendeley is not a search engine. Instead, it provides tools that can be used in conjunction with an academic search engine. It provides users with features to manage references, manage and showcase their research, and connect with other researchers [5].

6. Semantic Scholar: Semantic Scholar is a search engine that uses natural language processing and artificial intelligence in order to assist in the searching of papers. It was launched in November 2015, and has since added over 10 million papers into the system. They originally focused only on computer science papers and have recently expanded to neuroscience [6].

3. OUR SOLUTION

After researching the existing solutions, we have come up with an innovative way to improve the experience of accessing relevant academic papers. Our idea is simple: users will search by concepts instead of by content text. In addition, users will be capable of voting on those search results, resulting in real-time learning. Over time, this will gradually home in on the results that are truly relevant to each individual user.

Our search system has 4 major components: the loader module, the AI for concept extraction, the voting algorithm, and the session management mechanism.

3.1 Loader Module

To create a knowledge base of research papers, the foremost important aspect is to have a mechanism of discovering and indexing research papers. The best method we found was to implement a web crawler that discovers papers as it crawls over the internet. The web crawler would then index those papers into a knowledge repository. In the end, we found that this approach would have been time-consuming to develop, with a high chance of errors.

Instead, our approach is to use pre-existing repositories of research papers, extracting papers directly from it. Our loader module benefits by acting as an eager mechanism for finding papers. We are currently only using the Springer repository, due to their easy-to-implement API [7]. After papers are pulled, the loader sends the results to IBM Watson to extract concepts. [8]

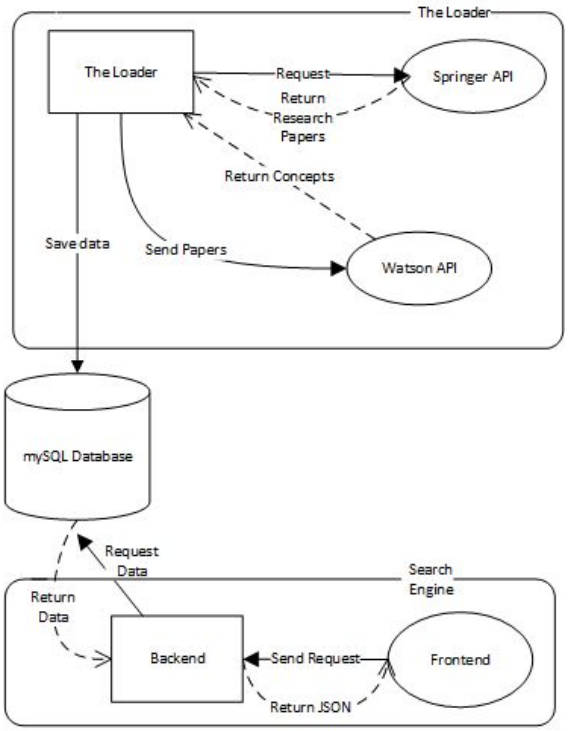
3.2 Concept Extraction

Instead of searching raw text, we want to search by *concept*. The contextual concepts of the paper therefore must somehow be extracted. Theoretically, this can be developed by using large sources of concepts (DBPedia, Yago, and Freebase to name a few), and then calculate the graph distances between the paper's text and each concept [9].

In the interest of time and complexity, we decided to implement IBM Watson's AlchemyAPI, instead. Alchemy analyzes the text of the paper and gives us back high-level core concepts, which we can then add into our relational database.

3.3 Voting Mechanism

When a user feels that a paper is particularly relevant or non-relevant to their search, they may upvote or downvote it. When a vote is registered, it is saved to the user's session; on subsequent searches, those user preferences will be factored in via the weighting algorithm described below.



1. The system pulls all papers that exist for the existing user's search query and stores them in a list.
2. The algorithm then pulls all concepts that are attached to all papers that were voted on by the user in their current session.
3. Pulls all the concepts for the papers from the search query along with the relevance of that concept in the paper.
4. It then processes each concept from the search query papers, it has 4 possible options for weighing in on a paper's concepts.

    x = The relevance of a paper's concept

    Rc = relevance of concept

    Vt = Summation of votes for a concept.

    Note: Vt can be value can be negative.

a. The concept could be part of the original search concepts and also exist in the papers that were voted on.

$$x = \frac{Vt}{4} * (Rc) + (Rc)$$

b. The concept could be part of the original search concepts, but does not exist as a voted on concept.

$$x = (Rc)$$

c. The concept could not be part of the original search concepts, but exist as a voted on concept.

$x = \frac{Vt}{4} * (Rc)$

d. The concept could not be part of the original search concepts and does not exist as a voted on concept. $x = 0$

5. It does a summation of the paper's concepts.
6. Normalize all the search query papers so they fall between 100% and 0%.

Pc = Paper's current relevance

Pf = Paper's final relevance

min = Lowest relevance of all the papers

max = Highest relevance of all the papers

$Pf = (Pc - min) / (max - min)$

This algorithm increases or decreases the priority of all the concepts related to the paper that was voted on, and refreshes the results on the screen to reflect those changes. The user now gets a result set which is more relevant to their search. In addition to the up and down voting, a user can save a paper. The algorithm takes saving a paper as an indication that the user would like similar papers and gives those concepts an extra boost.

3.4 Managing Sessions

To enable the application to remember what it learned about the user's requirements in a particular search session, we store each search query by each user as a separate session. This approach allows for multiple users to upvote, downvote, or save papers, but for the application to continue maintaining isolation between each user's preferences. Even when a user is not logged into the system, a separate session for that user is created to manipulate their searches.

4. FUTURE WORK

This prototype application has been developed in a mere two-month timeframe. There is still much to be done to improve the functionality and make it more user friendly. The next areas of development are as follows:

1. Implement filters to make the application more user friendly. The application currently only allows searches using a text search query, and there is no way to filter the search results using attributes like author name, date range, or journal name.
2. Implement pagination mechanism.
3. Allow users to edit their past sessions or to start where they left off.
4. Allow users to upload their paper and have our service pass it through IBM Watson. It will then pull out concepts, and do a search using those concepts over the database.
5. Allow a user to share a "learned" session with other users. Currently, if a user manages to teach the application what is being looked for, there is no way for it to be shared with another user, because it is saved in their local session..
6. Allow users to open papers on our service and color code it for faster reading.
7. Implement a forum system that allows users to discuss the topics that they are researching.
8. Use AI and natural language processing to convert search queries into sets of relevant keywords.
9. Incorporate analytics. This would greatly aid optimization of the core algorithms.
10. Design a mobile page for browsing on a mobile device.

6. ACKNOWLEDGMENTS

This application and paper was created through the guidelines provided by our instructor David Joyner, and with the support and mentorship of Phillip Ratif.

7. REFERENCES

[1] Summann, Friedrich and Norbert Lossau. 2004. "Search Engine Technology and Digital Libraries: Moving from Theory to Practice," D-Lib Magazine, Volume 10, Number 9, September 2004. doi:10.1045/september2004-lossau

[2] Péter Jacsó, (2011) "The pros and cons of Microsoft Academic Search from a bibliometric perspective", Online Information Review, Vol. 35 Iss: 6, pp.983 - 997

[3] Orduña-Malea, E.; Ayllón, J.M.; Martín-Martín, A.; Delgado López-Cózar, E. (2014). *About the size of Google Scholar: playing the numbers.* Granada: EC3 Working Papers, 18: 23 July 2014. ArXiV eprint

[4] *About*. (2016, 12 11). Retrieved from EBSCO: https://www.ebsco.com/about

[5] Hull, D.; Pettifer, S.; Kell, D. (Oct 2008). McEntyre, Johanna, ed. "Defrosting the digital library: bibliographic tools for the next generation web". doi:10.1371/journal.pcbi.1000204.

[6] *Frequently Asked Questions - Semantic Scholar.* (2016, 12, 11). Retrieved from Semantic Scholar: https://www.semanticscholar.org/faq

[7] *Springer | BioMed Central API Portal*. (2016, 12, 11). Retrieved from Springer: https://dev.springer.com/

[8] Watson Concept Insights: A Conceptual Association Framework. (2016). *World Wide Web*, (pp. 179-182). Montreal.

[9] Tsang, V., & Stevenson, S. (2004, May). Calculating Semantic Distance between Word Sense Probability Distributions. In CoNLL (pp. 81-88)

[10] Jöran Beel and Bela Gipp. Google Scholar's Ranking Algorithm: An Introductory Overview. In Birger Larsen and Jacqueline Leta, editors, Proceedings of the 12th International Conference on Scientometrics and Informetrics (ISSI'09), volume 1, pages 230–241, Rio de Janeiro (Brazil), July 2009. International Society for Scientometrics and Informetrics. ISSN 2175-1935.