Abstract

With the increased availability of large amounts of texts in recent years (thanks to popularity of the Internet and modern search engines)[1] automatic text summarization has emerged as one of the techniques that can help users make sense of information overload. Automatic summarization is the process of reducing a large amount of text to a smaller one while minimizing the information loss. Our work introduces the novel advanced summarization scenario of "query session guided multi-document summarization" (QSMDS). We created a dataset of human annotated query session guided summaries (focused on the medical domain) to allow the evaluation of automatic QSMDS systems. We also adapted state of the art automatic summarization methods to our scenario to achieve baselines on the task.

[1] http://www.internetworldstats.com/emarketing.htm

Acknowledgments

I wish to acknowledge my advisor, Prof. Michael Elhadad, for his guidance and ability to make sense of the clutter in my head. I also want to thank my lab partners Raphel Cohen (for all the coffee), Dr. Yael Netzer and Dr. Meni Adler for their constant help and attentiveness (and annotations). All of our annotators Sagit Fried, Inga Isakov, Noa Bakoff, Omer Basha, Hayon Avi, and Noam Roth. The Past two years in the BGU NLP lab were both fun and interesting.

Contents

List of Figures

Chapter 1

Introduction

In this chapter we explain the process that led us to build the "Query-Chain Focus Summarization Dataset" (QCFS). Since our objective is to combine work in Automatic Summarization and Information Retrieval, we first give a short overview of the field of automatic summarization, of the traditional Information Retrieval field, and specifically focus on the task of Exploratory Search. We then explain why QCFS is a unique summarization scenario that can greatly aid exploratory search systems.

1.1 Information Retrieval

1.1.1 Task

The task of information retrieval (IR) is to supply a user with information relevant to a given query. The query is usually a formal statement that should represent the information needs [1] of the user. An ideal IR system should be able to understand the information need behind the query and fetch only relevant data to the

need, *e.g., if the user wants to learn about* "the effects of red wine on the heart's health*", a document about a heart-shaped red wine bottle would match the query lexically but would not meet the information need.* Most IR systems assign to each document in its dataset a relevancy score[1] to a query and present the user with the highest scored documents.

1.1.2 Methods

Two well-known IR approaches are:

- **Vector Space Model [2]**- The query and the document are presented as vectors. Each dimension of the vector represents the TF-IDF[3] (this term will be explained later) of a specific word in the document or query. After encoding documents as vectors, algebraic methods such as vector Euclidean distance ($d(V,U) = \sqrt{(v_1-u_1)^2+(v_2-u_2)^2+...+(v_n-u_n)^2+}$) or cosine distance ($\cos\theta = \frac{U*V}{|U|*|V|}$) are used to determine the similarity of documents to the query.

- **Probabilistic Models [4]** - Given the query and document representations, a system has an uncertain guess of whether a document has content relevant to the information need. Probability theory provides a principled foundation for such reasoning under uncertainty.

1.1.3 Evaluation

Many different measures for evaluating the performance of information retrieval systems have been proposed. The measures require a collection of documents

[1]The score can be determined by the content of the document and their metadata

and a query. All common measures described here assume a ground truth notion of relevancy: every document is known to be either relevant or non-relevant to a particular query. In practice queries may be ill-posed and there may be different shades of relevancy.

- **Precision** - $Precision = \frac{|\{relevantdocuments\} \cap \{retrieveddocuments\}|}{|\{retrieveddocuments\}|}$ Precision is analogous to positive predictive value. Precision takes all retrieved documents into account.
- **Recall** - $Recall = \frac{|\{relevantdocuments\} \cap \{retrieveddocuments\}|}{|\{relevantdocuments\}|}$ Recall is often called sensitivity. So it can be looked at as the probability that a relevant document is retrieved by the query.
- **F-measure** - $F - measure = \frac{2 * precision * recall}{precision + recall}$ F-measure captures both precision and recall aspects.

Some of the better known datasets for evaluating IR systems are TREC[2] and MUC[3]

1.2 Exploratory Search

.

1.2.1 Exploratory Search in a Nutshell

As a part of the "Entailment-Based Exploratory Search and Summarization System For the Medical Domain" [5] project (a joint-project of Bar-Ilan and Ben-

[2] http://trec.nist.gov/data.html
[3] http://www-nlpir.nist.gov/

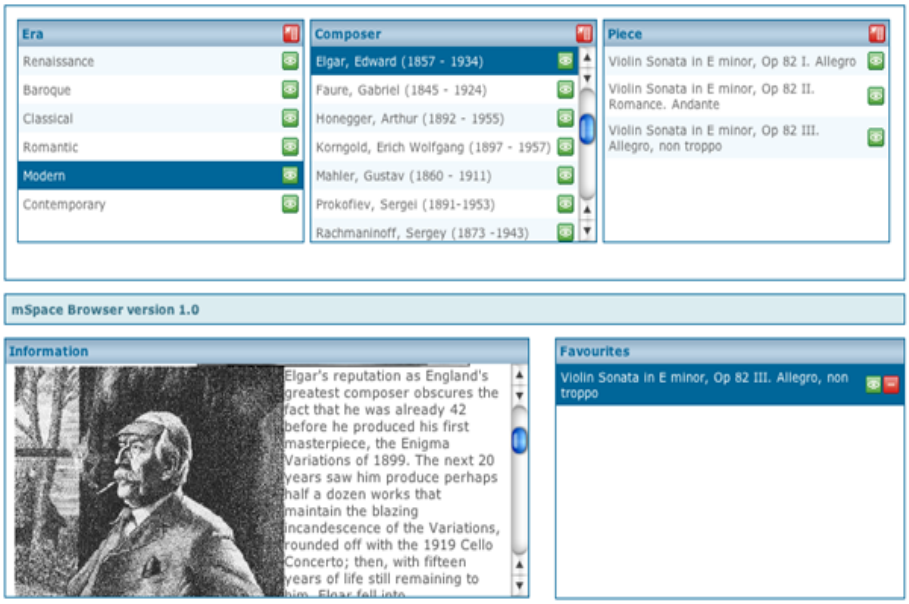

Figure 1.1: mSpace exploratory search system

Gurion Universities) we tried to confront the task of Exploratory Search [6], namely obtaining effective information from a corpus without prior knowledge of its content.

The task of exploratory search evolved from the fields of IR human-computer interaction (HCI). Exploratory search is different from regular IR tasks due to the following assumptions about the profile of the user using the system:

- unfamiliar with the domain of his goal
- unsure about the ways to achieve his goals
- or even unsure about his goals in the first place

These assumptions about the user mean that the user cannot formulate a query using appropriate search terms, because he does not know the terms of the domain he is exploring. Instead, different exploration methods must be designed. mSpace

is an early prototype that introduced human-computer interaction techniques that have since then be widely adopted to help users explore unfamiliar document collections. mSpace[4] is an exploratory search for classical music. Among its features are:

- **Multi-Faceted** - The mSpace Browser is a multi faceted column based client for exploring large data sets in a way that makes sense to you.
- **Multi-Media** - The mSpace browser takes advance of multi-media whenever possible. You could use audio samples to help explore unfamiliar areas of music, or video clips to aid with movie browsing.
- **Multi-Search** -As well as column based browsing, mSpace offers a powerful Search system. Try a simple keyword search or refine your query with built in Advanced Search.

A good exploratory search system should support[6]:

- **Querying and query refinement:**

  The user should be able to search the system using a query and each query should help him learn more about the domain and with the new knowledge he acquired he could refine his next queries.

- **Faceted search:**

  The facets are used to explore information by assigning multiple attributes to data. Each facet can narrow the search to display data with the specific attribute (of the facet). *e.g., most on-line retailers use facets such as price range, product color, product department (books, music, electronic...) etc.*

[4] `http://research.mspace.fm/mspace`

Those facets can be dynamically altered for the data displayed (color is not a relevant attribute facet when buying a book but it is a relevant attribute facet for buying a shirt).

- **Leverage search context:**

  The exploratory search system should be aware of the user's past search results to avoid displaying information the user already read.

1.3 Automatic Summarization

Automatic summarization is a field in natural language processing that involves reducing a text document (or documents) into a shorter summary using a computer program. The constantly increasing amount of textual information available to users on the Internet led to the development of many automatic summarization systems. Most of them are distinguishable by the following dimensions:

- **Informative vs. Indicative summaries:**

  An informative summary should capture all the information of the summary and could replace the need for reading the entire document. On the other end indicative summaries only help the user to decide if he wants to read the text. Indicative summaries are usually "snippets" of text associated with search results from information retrieval systems.

- **Single vs. Multi-document summaries:**

  A single document summary capture the information of a single document while a multi-document summary captures the information from a set of documents about the topics. When summarizing a document set it is easier

to find important information since the important information to the topic should appear in all of the documents while esoteric information should appear in only few documents. When summarizing a single document with no previous knowledge it is very hard to distinguish between the important information and esoteric information. When summarizing a single document it is easier to maintain coherence in the summary just by extracting sentences since all sentences share the same writing style.

- **Extractive vs. Generative summaries:**

  Extractive summaries construct the summary from sentences that appear in the original text. While generative summary extracts information from the original text and generates an entirely new text to summaries it.

The summarization task proved to be a difficult one because of the following reasons:

- **Detect Central Topics:**

  Automatic summarization systems should capture central topics from articles. Those topics might be mentioned only few times in the article. *e.g., an article discussing "phone call between Barack Obama and Hassan Rouhani" should not repeat the fact that the phone call was held more than once but it is expected to be mentioned in a summary of the article*.

- **Redundancy:**

  Salient segments coming from different documents often carry similar information, which is repeated in multiple documents. The generator must avoid including segments conveying the same information into the summary.

- **Coherence:**

  When segments are extracted from their source document, they may include references to textual entities within the source. Very little attention has been paid to this problem in practice.

1.3.1 Advanced Summarization Scenarios

In recent years shared tasks (DUC 2001-2007 and TAC 2008 and later[5]) presented advanced summarization scenarios tasks. The following tasks were relevant to our research:

- **Query-Oriented Summarization (QS)[7]:**

  The QS tasks aims to extract a summary from a set of documents, given a query. The summary should represent the most relevant[6] information to the query from the documents set. The QS task seeks to improve IR tasks. In most current search engines, a user usually enters a query and receives a large amount of search documents that usually match the query. QS is aiming to both return a single document that summarizes this set of documents and semantic information to the IR process.

- **Update Summarization (US)[8]:**

  The purpose of the US task is to distinguish between novel and redundant information in two document sets (documents A and documents B). The documents usually cover the same event but documents A temporally precede documents B (in the date it was written). The goal is to extract all

[5]Data Understanding Conference and since 2008 Text Analasys Confrences are annual confrences of large-scale evaluation of NLP methodologies.

[6]Relevancy of documents is determined to information need and not to a query

novel information from documents B into a summary and to avoid redundant information that the user is familiar with after reading documents A.

1.3.2 Summarization Evaluation

Evaluating summarizers is challenging because different people summarize the same input documents in different ways. Humans also do not readily agree on ways to evaluate the quality of human-written summaries. Over the years, the research community has converged on pragmatic methods to evaluate summarizers - both in a manual and automatic manner.

Manual Evaluation

Automatic summarizations can be evaluated manually using qualitative methods. In typical evaluations, each evaluator is presented with a questionnaire about the automatic summaries and the answers should help evaluate the summary. This questionnaire includes questions such as “How informative is the summary?”, “Is the summary coherent?”,“ How much redundant data does the summary contain?”, ... Another manual evaluation method is the Pyramid method [9]. Each of the model summaries are manually analyzed into meaning units usually corresponding to clauses, called basic elements. Automatic summaries are tested by checking whether they include the basic elements of the models. Overall linguistic quality of the summaries is also judged using qualitative questions. Such manual evaluation turns out to be consistent across human judges across years.

Automatic Evaluation

One of the main challenges in text summarization is evaluating the automatically generated summaries. Manual evaluation is expensive, time consuming and inconsistent between different evaluators. From all those reasons the need to an automatic summarization evaluation scheme has risen. Nowadays the most popular automatic summary evaluation tool is ROUGE [10]. ROUGE is a set of metrics comparing between a set of ideal manually crafted summaries and automatically generated summaries. ROUGE metrics proved to be consistent with manual evaluations [11]. The most commonly used ROUGE metrics are:

- **ROUGE-N: N-gram Co-Occurrence Statistics:**

  The ROUGE-N compares the recall of n-grams between the manually crafted summaries and the automatically generated ones.

  $$ROUGE - N = \frac{\sum_{S \in ManualSummaries} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in ManualSummaries} \sum_{gram_n \in S} Count(gram_n)}$$

- ROUGE-S: Skip-Bigram Co-Occurrence Statistics

  The skip-bigram is any per of words that appear in their sentence order allowing arbitrary gaps. ROUGE calculates recall precision and F_measure.

  $$R_{skip2} = \frac{SKIP2(X,Y)}{C(m,2)}$$
  $$P_{skip2} = \frac{SKIP2(X,Y)}{C(n,2)}$$
  $$F_{skip2} = \frac{(1+\beta^2)R_{skip2}P_{skip2}}{R_{skip2} + \beta^2 P_{skip2}}$$

  where m is the length of X and n is the length of Y, $SKIP2(X,Y)$ is the skip-bigram matches between X and Y, β controlling the relative importance of R_{skip2} and P_{skip2} ($\beta = 1$), C is the combination function

In order to evaluate our QCS system using ROUGE we needed to create a dataset of manually crafted query-chain focused summaries. In the next chapter we will explain how we created such a dataset.

1.4 The "Entailment-Based Exploratory Search and Summarization System For the Medical Domain" System

The "Entailment-Based Exploratory Search and Summarization System For the Medical Domain" is a collaborative effort of both Bar-Ilan and Ben-Gurion universities to create a exploratory search system for the medical domain. The objective of the research is combine insights from Entailment-based semantic analysis of text and automatic summarization to help users explore a document collection.

The overall strategy followed in the project is the following: with the help of textual entailment tools [12], a concept graph is generated from a large set of documents in the medical domain. The concept graph describes textual entailment relations between *propositions*. In this setting, a proposition consists of a predicate with two arguments that are possibly replaced by variables, such as ($X\ control\ asthma$). A graph that specifies an entailment relation ($X\ control\ asthma \rightarrow X\ affect\ asthma$') can help a user, who is browsing documents dealing with substances that affect asthma drill-down to only substances that control asthma.

In exploratory search terms, this technology provides an automated way to extract relevant facets at each stage of an exploratory session.

The user enters a query using free text and then can explore the relevant nodes

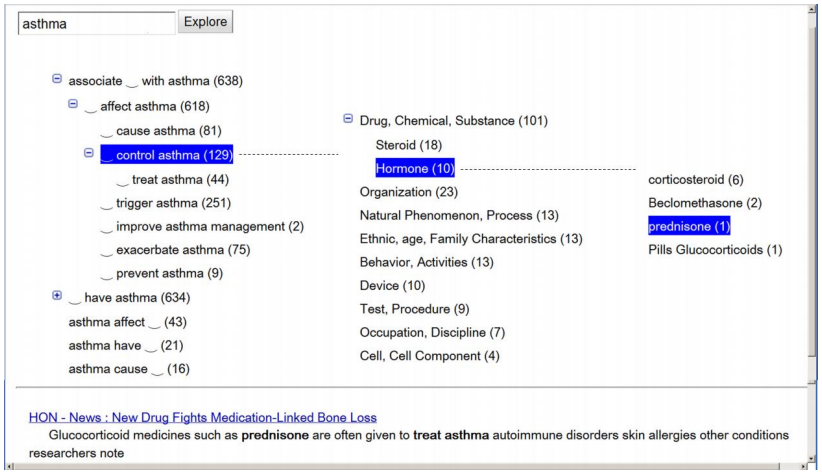

Figure 1.2: Example of the Entailment-Based Health Care Exploratory System

in the concept graph. When the user selects a facet, a subset of the documents is presented, and a summary is automatically generated for the focused documents. Since we assume that the users want to explore the dataset, we assume that several consecutive exploration steps will be taken and for each step, different summaries will be presented to the user. We want to avoid redundancy in the summaries presented to the user in the same exploration session, we want summaries from different exploration steps to present novel information. Our task is to build such a summarization system to aid the exploration process. This specific scenario is a new specialized summarization task - it resembles both Query-Focused summarization (because we want summaries to be relevant to the current query) and Updated Summarization (because we want successive summaries to be non-redundant). It is also different from each of these existing tasks: in contrast to Update Summarization, the set of past/new documents is not explicitly presented at each step of the exploration, and it must be constructed by the system; in contrast to Query-focused Summarization, the summary given at stage 2 of the

exploration must take into account that an answer to q1 was already provided.

In the rest of the work, we present the methodology we used to gather a dataset capturing such an exploratory behavior and the corresponding summaries produced by domain-experts. We then present algorithms and methods to model the desired behavior.

Chapter 2

Research Objectives

We formulate here the research questions addressed in this work:

- **Can we use automatic summaries to improve the exploratory search process?**

  Most exploratory search systems today include a combination of queries and facets that help the user discover documents that satisfy his information need. We want to improve this process by adding automatic summaries of the documents presented to the user in each step of the exploratory system. Those summaries should both capture the documents presented to the user and do not repeat information from previous steps in exploratory search process.

- **Do previous summaries effect the current summary?** We wish to research whether summaries presented to a user for the same query at different steps of the exploratory search process contain different content.

- **Can we use existing automatic summarization method for our task?**

  Both Query-focused Summarization (QS) and Update Summarization (US) tasks capture important aspects of the summarization task we have defined. QS should help us capture the query and facet aspects of our task, and US should help us capture the user learning process, since we want to avoid displaying data that the user has already learned from previous queries. The new task we introduce (query chain focused summarization) combines the tasks of QS and US. It is however different from both. We want to investigate whether a simple combination of the methods used for QS and US is sufficient to address the need of QCFS, or in contrast whether a new model is required.

- **What dataset can be used to evaluate QCFS algorithms?**

  DUC and TAC datasets contain both QS and US datasets and TREC contains an US dataset called Temporal Summarization. Since none of the available datasets capture both aspects of QS and US but only one for each dataset, we must create a new dataset. We address the method to gather such a new dataset in the next chapter.

Our main objective is to find an automatic summary method to integrate within an exploratory search system and to evaluate its performance against human observed performance.

Chapter 3

The Query Chain Dataset

In this chapter, we introduce the query chain dataset that we built in order to evaluate exploratory search summaries. We outline the content of the dataset, provide background information about the participants who contributed to the construction of the dataset, the technologies we used to build the dataset, and how we verified that the dataset matched our task.

3.1 Requirements on the Dataset

The dataset should capture summaries generated to aid in an exploratory search process. In order to gather such a dataset, we need to reproduce an exploratory search process, in which a user actively seeks to understand a new domain, and gather manually crafted summaries, produced by domain experts, and that address the information need of the user at each step of his exploration. We decided to focus on the consumer-health domain, in which users who are not medical experts seek to gather information about medical topics. For example, patients or their

relatives often seek to understand more about their condition. We also had access to the expertise of students in medicine to help us gather a proper set of documents for specific medical domains and produce appropriate answers to a set of query chains.

3.2 The Dataset Description

We first aimed to find real-word exploratory search sessions in the medical domain. Since no public exploratory search system exists, we decided to inspect query logs of an existing traditional search engine, and manually extract query chains that seemed to have exploratory features such as query refinement. We used the free medical database PubMed[1] query logs. PubMed contains a large collection of scientific articles in the Medical domain. It is usually accessed by researchers and medical professionals who are seeking specific articles in specific domains, using well targeted keywords. It is not a consumer-health site, such as WebMD or BeOK[13].

We found, however, that many non-professional users still access PubMeb to gather information. We identified such non-professional queries by looking for queries that include non-professional terms, that are quite general – for example, short queries such as ”asthma” or ”cancer”. Such short queries almost certainly identify ”naive users”.

Given these starting points, we then investigated the search log session for these users, and collected sessions that started from a general query term, rapidly followed by variations of the original query with some additional refinement terms.

[1] `http://www.ncbi.nlm.nih.gov/pubmed`

A typical example of such a search session is the sequence of queries: *asthma* → *asthma causes* → *asthma allergy* → *asthma mold allergy*

We found many such examples of query chains in the PubMed query logs, and focused on query chains around the topics of 'asthma' and 'lung cancer'. We specifically collected the following chains from the PubMed query logs:

- Asthma:
  - asthma causes → asthma allergy → asthma mold allergy
  - asthma treatment → asthma medication → corticosteroids
  - exercise induced asthma → exercise for asthmatic
  - atopic dermatitis → atopic dermatitis medications → atopic dermatitis side effects
  - atopic dermatitis → atopic dermatitis children → atopic dermatitis treatment
- Lung cancer:
  - Lung cancer → Lung cancer causes → Lung cancer symptoms
  - Lung cancer diagnosis → Lung cancer treatment → lung cancer treatment side effects
  - Stage of lung cancer → Lung cancer staging tests → Lung cancer TNM staging system
  - Types of lung cancer → Non-small cell lung cancer treatment → Non-small cell lung cancer surgery

- Lung Cancer in Women → Risk factors for Lung Cancer in Women → Treatment of Lung Cancer in Women
- Lung Cancer chemotherapy → Goals of Lung Cancer chemotherapy → Palliative care for Lung Cancer

After we obtained the queries, we needed to obtain documents for our simulated exploratory search system. We searched the Internet for documents that correspond to each query, in sites with high-prestige such as: "Asthma and allergy Foundation of America," "Asthma New-Zeland," "Cleveland Clinic," "about.com," "United States Environmental Protection Agency," "health.com," "lifescript.com," "NIH Heart," "Lung and Blood Institute," "National Library of Medicine - National Institutes of Health," "Palo Alto Medical Foundation," "Webmd", "Wikipedia," "lungcancersurgery.org," and more. The guidelines when gathering data for the document set were to be as comprehensive as possible (to make sure all of our queries will be covered by the documents) and also insert some redundancy to the document set (in order to make sure the Update Summary part of the system will be tested). Overall, we collected 125 documents about "asthma" and 135 documents about "lung cancer".

	Documents Count	Sentence Count	Word Count	Unique Words
Documents (asthma)	125	1,924	19,662	2,284
Documents (cancer)	135	1,450	17,842	2,228
Documents (total)	260	3,374	37,504	3,399
Queries (asthma)	15	15	36	14
Queries (cancer)	18	18	71	25
Queries (total)	33	33	107	37
Manual Summaries (asthma)	45	543	6,349	1,011
Manual Summaries (cancer)	54	669	8,287	1,130
Manual Summaries (total)	99	1,212	14,636	1,701

3.2.1 The Annotators

Five people participated in the summarization of the dataset: a linguistics MSc student, a medical student, a computer science MSc student, medical public health MSc student and a professional translator with a doctoral degree with experience in translation and scientific editing. All summarizers had very good English skills.

For each query, annotators manually created a summary, overall 3 summaries per query. In addition, the annotators selected key sentences for each summary, in order to enable alternative evaluation methods beyond ROUGE. We asked the annotators to view each step of the query chains one by one, then to answer the query by searching the document collection, selecting sentences appropriate to the query step, then rephrasing the summary on the basis of the selected sentences.

The summarizers were instructed to not include redundant information in the same query chain.

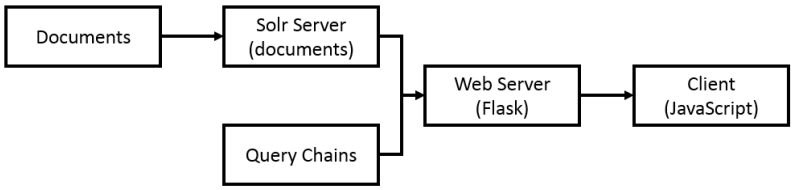

Figure 3.1: Our annotation site architecture

The summarizers reported that an average query summarization took about half an hour.

For each query, we used the three summaries with the highest agreement score, as measured by the ROUGE scores.

3.2.2 Technology Review

We created a web site[2] to help the summarizers write their summaries. The website allowed navigation of both the queries and the document set. The queries were navigated using a forward and backward buttons and the dataset was indexed using "Apache Solr" (an open source enterprise search platform from the Apache Lucene^{TM} project)[3]. The summarizers were presented with the query chain and could use a free-text search to retrieve documents relevant to the query (using Solr default TF/IDF search). After a summarizer selected a document, he was presented with the document content split by sentences (this was achieved with the NLTK package[4] $sent\_tokenize$). The summarizer could then select each sentence and mark it as relevant to the query and copy it to the summary. The site

[2] http://www.cs.bgu.ac.il/~talbau/static/login.html
[3] http://lucene.apache.org/solr/
[4] http://nltk.org/

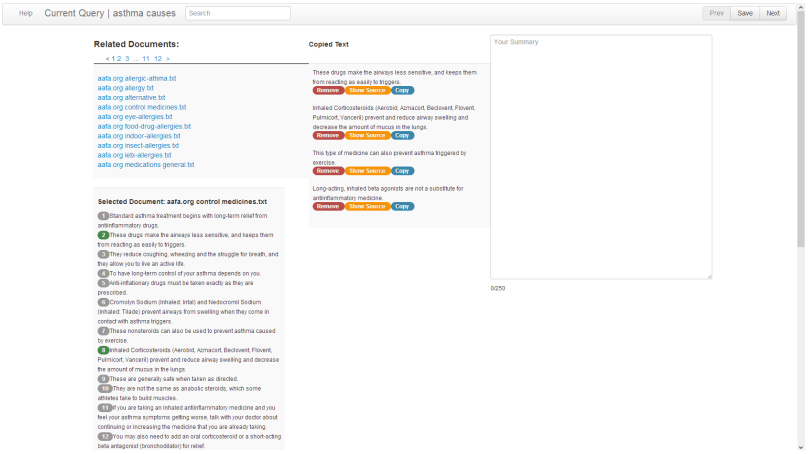

Figure 3.2: Our annotation site interface

was created using JavaScript[5] and HTML5[6] on the client side and Python Flask[7] on the server side.

3.2.3 Verifying the Dataset

To verify our dataset, we needed to make sure that the summaries we created for advanced queries (q2, q3 in each chain) are different from the summaries created for the same queries by summarizers who did not see the previous summaries in the chain. We asked from additional annotators to create manual summaries of advanced queries from the query chain without ever seeing the queries from the beginning of the chain. We collected 15 summaries, 3 for each 2nd query from the asthma query chains.

We first tested the mean ROUGE score of second query summaries. ROUGE compares a summary with a set of reference summaries. The mean ROUGE score

[5] http://jquery.com/
[6] http://getbootstrap.com/2.3.2/
[7] http://flask.pocoo.org/

is the mean score of each manual summary vs. all other summaries about the same query. We got the following results: $r1 = 0.52$, $r2 = 0.22$, $rs4 = 0.13$. The mean ROUGE scores of the second query summaries by people who did not see the previous query were ($r1 = 0.49$, $r2 = 0.22$, $rs4 = .01$).

We only verified the asthma dataset in this manner The results, except for the R2 test, had statistically significant difference with 95% confidence interval, and indicate that summarizers produced markedly different summaries in the context of an exploratory search from those produced for the same query out of context.

More information on the dataset could be find at:

`www.cs.bgu.ac.il/~talbau/QSMDS/dataset.html`

Chapter 4

Methods

In this chapter we will describe some of the methods we created for the task of adding summarizations to the BIU exploratory search system. We created two naive baselines and three adaptations of state-of-the-art automatic summary methods that adds both query and context features to those methods. The input for those methods is a document set and query chain and output is a summary for each query in the chain that best answers that query in context of the chain.

4.1 Naive Baselines

First we tested two very naive baselines for our task. These methods do not address the query chain aspect of the problem. We included those baselines in this work to help the reader get a better perspective on ROUGE scores. The first naive baseline we tested was the 'First Sentence Method', In this method we ranked all the documents in the dataset by their TF/IDF score for the given query, then we took the first sentence of each document till we reached 250 words. This methods

assumes the first sentence is usually the most important sentence in each document (this assumption is more relevant when summarizing news articles). The second naive method we tested was the 'First Doc Method' which is extracting the first 250 words form the documents with the best TF/IDF score for the current query. This method solves a well known problem in the multi-document summarization field, the coherence problem. Usually when extracting sentences from different documents they tend to be very incoherent and by taking the entire document we achieve coherence in our summary but with the price of not covering all the information.

4.2 LexRank Based Methods

LexRank [14] is a graph based algorithm for computing the relative importance of textual units which as been used for text summarization to identify the centrality of sentences. In this section we will explain how the algorithm works, and how we modified it for our task.

LexRank creates the following weighted undirected graph from the given document set. Each node in the graph represents a sentence from the document set. For each node, the sentence is encoded as word-vector. For each sentence we save the TF/IDF (term frequency/inverse document frequency) [3] value in the appropriate coordinate of the vector. $IDF_i = \log(\frac{N}{n_i})$ N is total number of documents and n_i is the number of documents containing the word i. Each edge weight is determined by the cosine distance of the vectors we created.

$$CosineDistance(A, B) = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt[2]{\sum_{i=1}^{n} (A_i)^2} \times \sqrt[2]{\sum_{i=1}^{n} (B_i)^2}}$$

After the graph is created we use PageRank [15][16] to rank the nodes (sentences) by their centrality in the graph we created. PageRank is a link analysis algorithm, named after Google co-founder Larry Page. It is designed to be used by the Google web search engine to rank sites by importance. For LexRank's case, we use the algorithm to find central sentences. Consider a graph where each node represents a site and an edge weight is the probability that a user browsing the site will go from that site to the other site. This probability is estimated by the number of hyper-links from this site to the other site plus the damping factor. The damping factor is the probability that a user will click the address bar and navigate to a new random site. Each node P_i is assigned a Rank $PR(p_i)$ by the following formula.

$$PR(p_i) = (1-d)\frac{1}{N} + d \sum_{p_j \in Neighbors(p_j)} \frac{PR(p_j)}{|Neighbors(p_j)|}$$

N is the size of the graph, $Neighbors$ is a function that maps each node to the nodes connected to it, and d is the damping factor. This definition is recursive. There are three common methods used to solve this problem an iterative method, random walk, and an algebraic method.

- **Algebraic method** - We can write all the edges values in a matrix A (an $n*n$ matrix) where the value of cell A_{ij} is the probability to move from site i to site j (also the value if the edge ij). Given a rank vector $\vec{V}$ exists[1] (an n sized vector where $\vec{V_i}$ is the rank of i). We now observe the multiplication $\vec{V} * A$ we notice that the value of each cell in the output vector is the the definition of the rank[2]. Then $\vec{V} * A = V$ and $\vec{V}$ is an eigenvector of A

[1] proof to the existence of this vector can be found in the referenced PageRank paper
[2] after including the damping factor in the probability

with the eigenvalue of 1. In the algebraic method we can simply search an eigenvector of A with eigenvalue of 1.

- **Random walk [17]** - We can estimate the rank of each node in the graph with a process called random walk. We start at a random node on the graph and then randomly move to another node with the probability given by the edges. After enough iterations the rank of each node is determined by the number of times the algorithm reached the node divided by number of iterations.

- **Iterative method** - The iterative method assigns an initial rank to each nodes (usually $PR_{t0}(i) = \frac{1}{N}$) and at each step applies the rank formula $PR_{t+1}(p_i) = (1-d)\frac{1}{N} + d\sum_{p_j \in Neighbors(p_j)} \frac{PR_t(p_j)}{|Neighbors(p_j)|}$ till the rank converges or $|PR_{t+1} - PRt| < \epsilon$.

We used the iterative method for our experiments.

Now that we have a rank assigned to each sentence, we can select the top ranked sentences to form our summary. In order to avoid redundancy we select a threshold and do not add sentences that are too similar to sentences already selected.

Data: Document Set D,Threshold T,wordLimit

Result: Sentence Set S

Nodes = {};

TFIDF = calcTFIDF(D);

for *sent in D* **do**

    Nodes.add(TFIDF.getvector(sent));

end

E = map node pair to score;

for $Node_1$ *and* $Node_2$ *in Nodes* **do**

    E$\{Node_1\ Node_2\}$ = CosineDistence($Node_1\ Node_2$);

end

Rank = PageRank(V E);

while *True* **do**

    newSent = Rank.pop();

    for *sent in S* **do**

        if *E(newSent sent)* $\leq$ *T* **then**

            S.add(newSent);

        end

        if *countWords(S)* $\geq$ *wordLimit* **then**

            return S;

        end

    end

end

Algorithm 1: LexRank outline

4.2.1 Modified LexRank

We modified LexRank to handle query-oriented summarization and to better adapt it to the medical domain.

Work done by Miao, Yajie and Li, Chunping on "Enhancing Query-oriented Summarization based on Sentence Wikification"[18] inspired us to add semantic features to the graph (plain LexRank weight is determined only by lexical features). To achieve semantic weight, we extracted two types of terms from each sentence Wikipedia terms using Illinois Wikifier [19] and UMLS (Unified Medical Language System) [20] terms using HealthTermFinder [21]. Once the terms obtained we used Wiki-Miner [22] to compute the similarity score between two Wikipedia terms and Ted Pederson's UMLS::Similarity [23] for the similarity score between two UMLS terms.

Now that we obtained the terms pairwise score we need to determine the similarity score between the sentences (each sentence contains many terms). To achieve this score we used Lexical Semantic Similarity (LSS) [24]

$$LSS(H,T) = \frac{\sum_{HW_i \in H}(\max_{TW_j \in T}(\frac{SSim(HW_i,TW_j)}{SSim(HW_i,HW_i)} * IDF(HW_i)))}{\sum_{HW_i \in H} IDF(HW_i)}$$

where $SSim$ is the similarity function we used (Wiki-Miner for Wiki-terms UMLS::Similarity for UMLS-terms) and IDF is Inverse Document Frequency as defined in the previous section. the weight of each edge will be a combination of similarity scores

$$E(U,V) = LSS_{lexical}(U,V) + A * LSS_{Wiki}(U,V) + B * LSS_{UMLS}(U,V)$$

e.g., in the figure we can see two sentences with zero lexical similarity but high semantic similarity[3] *that can be achieved with our term extraction method.*

We normalized the outgoing edges weight so that their sum is one (needed for

[3]LSS score was not computed for this example since IDF values were required

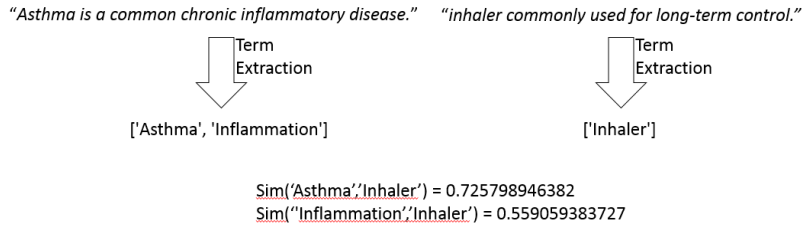

Figure 4.1: Term extraction and similarity example

the PageRank to converge).

In order to handle the query we added to the LexRank graph a new node representing the query and all the required edges. We change the page rank algorithm to achieve similarity [25] instead of just finding central sentences [26]. To do that we instead of the damping factor we added to each node a probability to return to query, instead of jumping to random node in normal PageRank settings. With this modification we achieve rank by similarity to the query because now instead of simulating a random surf we simulate the probability of reaching a sentence when starting a random walk at the query.

Since LexRank creates an exponential amount of edges we could not simple apply the algorithm to the entire document set. Instead we applied only on the top N Solr results for the query. We wanted to use this data for our sentence similarity measure.

Wiki-terms Similarity

Wikipedia is a collaboratively edited, free Internet encyclopedia it is one of the world's most popular general reference sites ever. Its huge amount of semi-structured data as recently became a popular source for Information Retrieval tasks. Most Wikipedia pages link to other related Wikipedia pages. This links

form what is know as the Wikipedia graph.

For our work, we used two state-of-the-art tools to extract the sentence similarity measures. First, we used the Illinois Wikifier [22] to extract the sentences and then Wiki-Miner [22] for terms similarity.

The Illinois Wikifier tries to solve the Disambiguation to Wikipedia (D2W) task by formalizing it as an optimization problem with local and global variants. The local variants assigns scores to Wikipedia term by the relatedness to each mention in the given text separately. The global variants search for coherence between the found terms. For example if 'Michael Jordan' refers to the page of the computer scientist rather than the basketball player the term 'Monte Carlo' in the document should be linked with the statistical technique rather than to the location.

The Wiki-Miner uses the Wikipedia graph to achieve similarity. The weight of each edge is determined by a TF/IDF score of links to that page and the similarity scores will be determined by the cosine distance of their hyper-link vectors.

UMLS Similarity

UMLS is a controlled vocabulary created and maintained by the US National Library of Medicine. The UMLS integrate and distribute key terminology, classification ,coding standards and associated resources to promote creation of more effective and interoperable biomedical information systems and services, including electronic health records.

To extract the UMLS terms from the free text we used HealthTermFinder, an automatic medical term annotation program that identifies UMLS terms in free text. The content of the notes was pre-processed to identify shallow syn-

tactic structure: part-of-speech tagging with the GENIA tagger [27] and phrase chunking with the OpenNLP toolkit[4]. HealthTermFinder recognizes named entities mentioned and maps them to semantic concepts in the UMLS. It was tested on a gold standard of 35 clinical notes from the Columbia University Medical Center. The notes contained 2,589 mentions of clinical entities, corresponding to 1,056 unique entities, as recognized through gold standard manual annotation. When compared with state-of-the-art MetaMap [19], HealthTermFinder identified the mentions with significantly better success: 88.55 (.013 95% CI) F-measure vs. 77.54 (.0165 95% CI) for MetaMap for exact matches of mentions.

For the similarity computation we used UMLS::similarity. UMLS::Similarity implements a number of semantic similarity and relatedness measures that are based on the structure and content of the Unied Medical Language System graph.

4.2.2 LexRank-Update

To add a previous queries to the modified LexRank model we incorporated the following changes to the model. first we did not create a new graph but merged the graph from the previous query and with the new query and sentences from the top N documents fetched. the edges were created using the same scoring as before. this process was done to enable a drill down effect to the summary. since the summarizer can access both sentences similar to the current query and from documents that are similar to the previous query.

Another modification to LexRank was done in the sentence selecting process (taking place after the ranking) we compared the highest ranked sentence to all sentences selected and sentences from previous summaries to avoid redundancy.

[4]http://opennlp.apache.org/

```
Data: Document Set D,Previous Graph G,Previous summary PS,Query
      Q,Threshold T,wordLimit
Result: Sentence Set S
for sent in D + Q do
    G.addNode(sent);
    for node in G do
        G.addEdge(node,sent,weight = LSS(sent,node));
    end
end
Rank = SimRank(G,Q);
while True do
    newSent = Rank.pop();
    for sent in S ∪ PS do
        if E(newSent sent) ≤ T then
            S.add(newSent);
        end
        if countWords(S) ≥ wordLimit then
            return S;
        end
    end
end
```

Algorithm 2: LexRank-Update outline

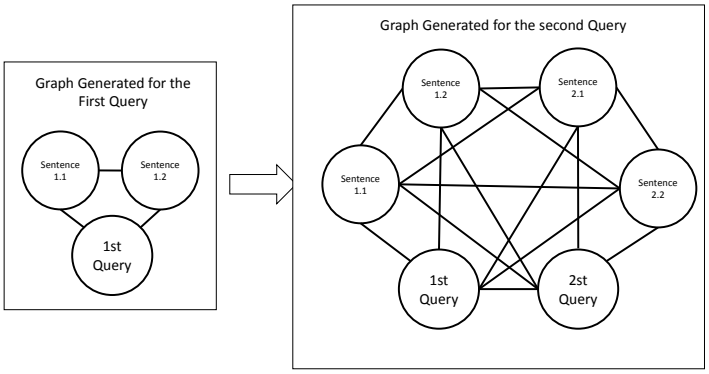

Figure 4.2: Graphs generated for LexRank Modified Update

4.3 KLSum Based Methods

The KLSum [28] algorithm was introduced by Haghighi and Vanderwende (2009) as a multi-document summarization model. The intuition needed to understand the algorithm is very straight-forward. We want to summarize a set of documents D by selecting a group of sentences S from our documents set that best describes the content of the documents set. In order to do that we create P_D the unigram distribution of D and try to find the group of sentences S that its unigram distribution P_S is most similar to P_D. We use $KullbackLeibler$ [29] divergence (D_{KL}) to evaluate similarity between the distributions.

$$D_{KL}(P||Q) = \sum_i ln(\frac{P(i)}{Q(i)}P(i))$$

KLSum finds the sentence set with minimal KL-Divergence to the original document set.

$$S = \min_{S:|S|\leq wordlimit} D_{KL}(P_D||P_S)$$

The algorithm complexity is $NP - hard$ (can be easily proved by a polynomial reduction from $SUBTSETSUM$). We used a greedy implementation of KL-Sum. Our implementation iterates over all the sentences in D and each time adds the the sentence that improves $D_{KL}(P_D||P_S)$ the most for the current iteration.

```
Data: Document Set D
Result: Sentence Set S
S = {};
while TRUE do
    best_improvement = MAX_FLOAT;
    best_sentence = NULL;
    for sent in D do
        if best_improvement ¡ KL(D,S + sent) then
            best_sentence = sent;
            best_improvement = KL(D,S + sent);
        end
    end
    S += best_sentence;
    if num_of_words(S) ¿ word_limit then
        return S;
    end
end
```

Algorithm 3: KLSum Greedy Implementation pseudo code

4.3.1 Our KLSum Implementation

Since the annotators used our annotation site to create their summaries we can safely assume most of them read documents in the top 10 results fetched by searching for the query. Thats why for creating the KLSum method we summarized only the top 10 sentences fetched from Solr using the given query. It is very important to mention that this method just answered single queries. It did not have any previous answer context.

4.3.2 KLSum-Update

For our second method we wanted to address the main problem of KLSum which is the lack of context. In order to add such context to the new model, we added the unigram distribution of previous summaries in the session to the current summary unigram distribution. When adding the previous unigram distribution we used the following smoothing:

- We only added words that appear in the current top N documents fetched by Solr to avoid division by zero when calculating KL-Divergence
- When adding the previous distribution we divided the number of instances by a smoothing factor (set to. Not using this factor, the previous summaries weight is too high and only esoteric sentences (in the sense that they contain words with very low frequency in the document set we want to summarize) are added to the summary.

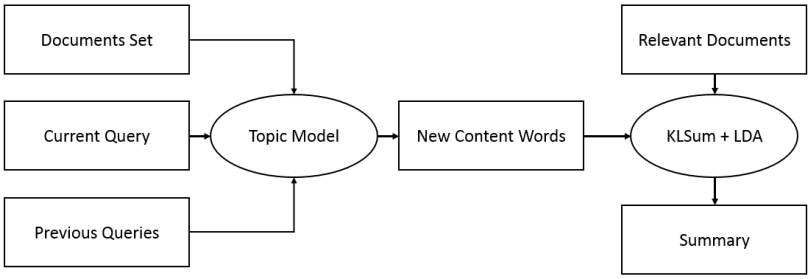

Figure 4.3: KLSum + LDA architecture

Unigram Distribution of Word X	
KLSum	KLSum Update
$\frac{\text{CountWordIn}(X, CurrentSummary)}{\text{NumOfWords}(CurrentSummary)}$	$\frac{\text{CountWordIn}(X, CurrentSummary) + \frac{\text{CountWordIn}(X, PreviousSummary)}{SmoothingFactor}}{\text{NumOfWords}(CurrentSummary)}$

4.3.3 KLSum with LDA

In this method, we also tried to address the lack of context in KLSum by using a topic model that tries to distinguish between old content originated from previous summaries and new content. Our topic model "Query Chain Topic Model" (see appendix) is based on Latent Dirichlet Allocation [30]. The Query Chain Topic Model can identify words appearances that contain content that is characteristic to current query. After we identified those words, we used KLSum to extract a summary. Instead of the regular unigram distribution we increased the probability of new content words.

$$P(X) = \frac{\text{CountWordIn}(X, CurrentSummary) + \text{ApperancesAsNewContent}(X)}{\text{NumOfWords}(CurrentSummary)}$$

4.4 Sentence Ordering

The algorithms listed above rank sentences by centrality and select items by top centrality, and in a way to avoid redundancy. We event determined so far how selected sentences should be ordered to produce a coherent summary, we used the following basic editing function for sentence ordering. We sorted the sentences by a lexicographical order, we first compared the TF/IDF score between the query and the documents that the sentence were taken from if they were equal, we ordered the sentences by their order in the original document.

Chapter 5

Results Analysis

In this chapter, we discuss the evaluation of our methods and the experiment we designed to assess our model.

5.1 Model Evaluation

5.1.1 UMLS and Wiki Coverage

For our Wiki and UMLS terms coverage test, we did not have a gold standard tagged dataset. We tried to find tagging errors in our dataset by manually checking terms with very low scores compared to other terms. Once we found such term we classified them into two types of errors:

- Wrong sense error: *e.g., the term 'Ventolin (e.p)' (a song by electronic artist Aphex Twin) was tagged instead of 'Salbutamol' a quick relief drug used for bronchospasm in conditions such as asthma and chronic obstructive pulmonary disease marketed under the name 'Ventolin'*. Those errors were

manually fixed by forcing the correct sense.

- Unfixable errors: *e.g., terms such as 'States and territories of Australia' found in the sentence "You also can look for asthma-related laws and regulations in each state and territory through the Library of Congress (see Appendix 5)."*. Those errors were manually fixed by not tagging any term.

We did not fix missing terms errors. In our document set (containing 37,504 words) we found 3,674 Wiki terms out of them 720 unique terms. We found 5,739 UMLS terms of them 1,245 unique terms.

5.2 Manual Evaluation

The manual evaluation was achieved with the help of two of the manual summaries annotators. We read and compared automatically generated summaries and tried to understand the differences between the methods. The first thing the annotators noticed was the low coherence of the summaries. Some summaries, were reported to be very informative. The second observation on the summaries was that all methods perform better the more specific the query is. For example for the first query chain *"asthma causes → asthma allergy → asthma mold allergy"*, summaries for the first query usually included sentences that just presented the subject of asthma causes. For example *Triggers are things that can cause asthma symptoms, an episode or attack or make asthma worse.* but did not give any information that could help the user learn about the query. For more specific queries such as *asthma mold allergy*, there was less information on the subject and, therefore, most of the information was relevant to the query. The biggest problem with

summaries that were generated for the advanced queries was redundancy, *e.g., in the summary generated by KLSum for "asthma mold allergy", we can observe most of the sentences are very relevant to the query but there are also sentences that include information from the previous query "asthma allergy" such as the emphasized sentence*.

It is possible to notice that the same query generated with the KLSum+LDA method does not include redundant information from previous queries in the chain.

We summarized the annotators comments and observations in the following table.

Method	Coverage[1]	Redundancy[2]	Comments
LexRank	medium	some[3]	a lot of lexical appearance of the query but not enough content.
LexRank Update	medium	some	the annotators could not notice the improvement in redundancy.
KLSum	good	noticeable	tendency to prefer longer sentences.
KLSum Update	good	good	summaries were noticeable less coherent.
KLSum+LDA	good	good	low coherence but better than the others.

Mold grows in humid, damp environments, so the best way to prevent and control indoor mold is to keep your home as dry and ventilated as possible.
Some people with mold allergies may have allergy symptoms the entire summer because of outdoor molds or year-round if symptoms are due to indoor molds.
You're at increased risk of getting these conditions, known as allergic fungal sinusitis and allergic bronchopulmonary aspergillosis, if you're allergic to mold.
Asthma triggers are different from person to person and can include: Airborne allergens, such as pollen, animal dander, mold, cockroaches and dust mites Allergic reactions to some foods, such as peanuts or shellfish Respiratory infections, such as the common cold Physical activity (exercise-induced asthma) Cold air Air pollutants and irritants, such as smoke Certain medications, including beta blockers, aspirin, ibuprofen (Advil, Motrin, others) and naproxen (Aleve) Strong emotions and stress Sulfites and preservatives added to some types of foods and beverages Gastroesophageal reflux disease (GERD), a condition in which stomach acids back up into your throat Menstrual cycle in some women.
Outdoor allergies (also called seasonal allergic rhinitis [SAR], hay fever, or nasal allergies) occur when allergens that are commonly found outdoors are inhaled into the nose and the lungs causing allergic reactions.
People may become allergic to only mold or fungi, or they may also have problems with dust mites, pollens and other spores.
Several measures will help: Stay indoors during periods when the published mold count is high.

Figure 5.1: KLSum summary for "*asthma mold allergy*"

Indoor allergies (perennial allergic rhinitis [PAR] or often called nasal allergies) occur when allergens that are commonly found indoors are inhaled into the nose and the lungs causing allergic reactions.
Several measures will help: Stay indoors during periods when the published mold count is high.
Some people with mold allergies may have allergy symptoms the entire summer because of outdoor molds or year-round if symptoms are due to indoor molds.
It can trigger asthma attacks and allergic reactions (such as hay fever or eczema), and it can even cause health problems in people without those conditions.
Mold grows in humid, damp environments, so the best way to prevent and control indoor mold is to keep your home as dry and ventilated as possible.
Asthma triggers are different from person to person and can include: Airborne allergens, such as pollen, animal dander, mold, cockroaches and dust mites Allergic reactions to some foods, such as peanuts or shellfish Respiratory infections, such as the common cold Physical activity (exercise-induced asthma) Cold air Air pollutants and irritants, such as smoke Certain medications, including beta blockers, aspirin, ibuprofen (Advil, Motrin, others) and naproxen (Aleve) Strong emotions and stress Sulfites and preservatives added to some types of foods and beverages Gastroesophageal reflux disease (GERD), a condition in which stomach acids back up into your throat Menstrual cycle in some women.
Pollen and Outdoor Mold What to do during your allergy season (when pollen or mold spore counts are high): Try to keep your windows closed.

Figure 5.2: KLSum+LDA summary for "*asthma mold allergy*"

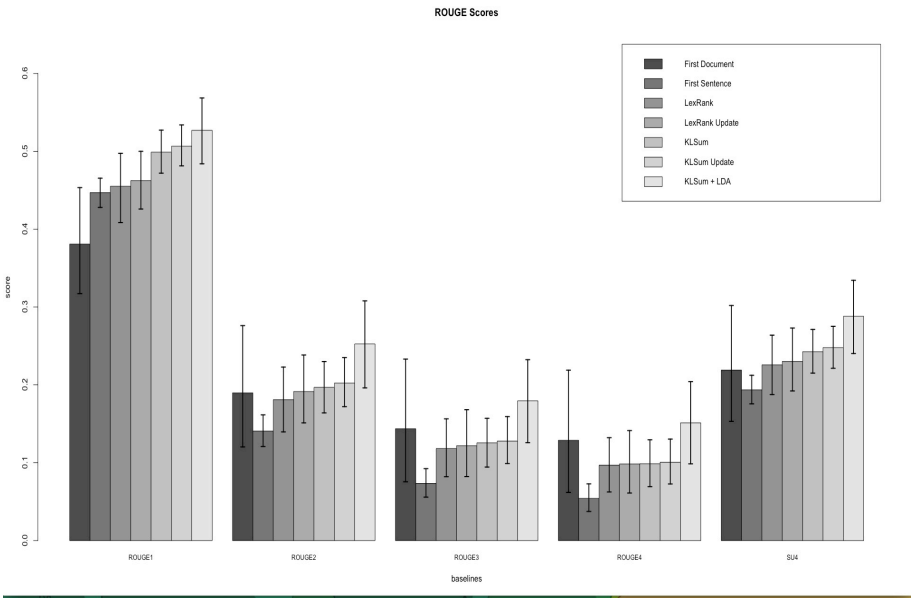

Figure 5.3: ROUGE Scores on the Full Queries Dataset

5.3 Automatic Evaluation

In this section, we present our automatic evaluation methods and results. We used ROUGE to automatically evaluate our summaries by comparing them to manually written summaries. We used ROUGE1, ROUGE2, ROUGE3, ROUGE4, and SU4. Most evaluation work today uses only ROUGE1, ROUGE2, and SU4 but recent reports show that ROUGE3 correlates best with human evaluation [11]. We tested our summaries vs. the summaries we created for our dataset and vs the summaries created by annotators who only saw advanced queries.

Observations from the automatic evaluation of full queries:

- KLSum based methods outperform the LexRank based methods: this is probably related to the poor coverage of relevant sentences reported in the manual evaluation.

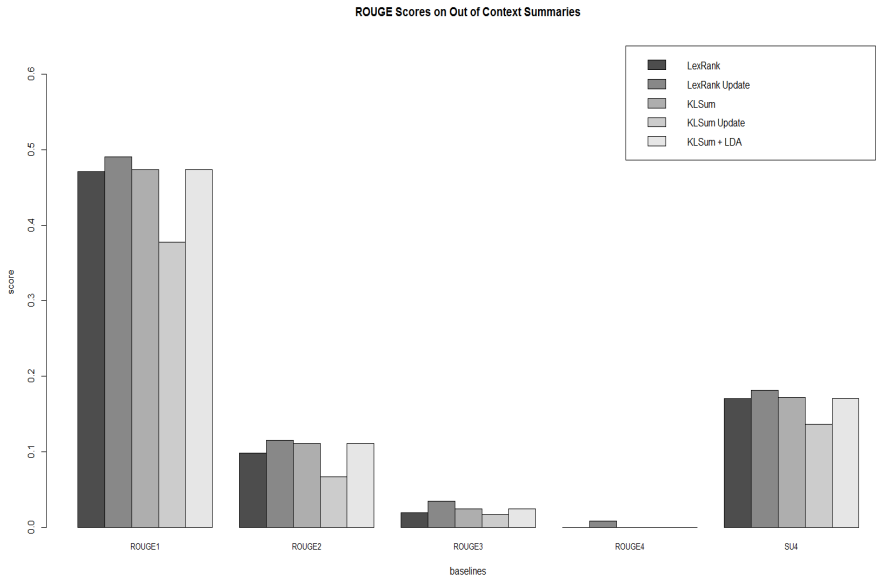

Figure 5.4: ROUGE Scores on Out-of-Context summaries

- KLSum + LDA is the best performing method but statistically significant to the second runner in only one test.
- Except for the case of the first document method there is agreement on the methods rank on all tests.

We also evaluated our methods on summaries created by annotators who only saw advanced queries in the chain[4]. Our assumption is that methods that use context from previous queries will perform worse (since they will ignore general content that should be included when the query is taken out of context).

Observations from the automatic evaluation of out-of-context queries:

- It is important to emphasize that this dataset is much smaller than the full dataset (3 queries vs 36 queries in the full dataset). In addition, the annotators for this dataset were all volunteers since we could not use the annotators

[4]This dataset is much smaller than the regular dataset and contains only three queries

for the full dataset since they all had seen the previous queries and might have been biased. We did not include error bars since for such a small dataset they are meaningless.

- The LexRank based algorithms did not perform as expected and LexRank Update performed better than unmodified LexRank.
- All KLSum based models that used context features preformed worse or the same as KLSum. as excepted.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this section, we return to our research questions and assess how we confronted them.

- **Can we use existing datasets to evaluate summarization methods for exploratory search?**

  Existing summarization datasets lack crucial features needed for our task: they are either generic, query-focused or update oriented – but none of them combines query-focused and query-update aspects. We created a new dataset to evaluate query-chain summarization. This dataset captures query answering and avoiding redundancy between exploration steps.

- **Can we use existing automatic summarization method for our task?**

  We created 3 novel automatic summarizations methods for the query-chain task. Those methods are based on existing state-of-the art methods. We

showed that by adding features related to task we can improve evaluation scores.

- **Do previous summaries affect the current summary?**

  We compared summaries created by annotators who summarized the entire query session and annotators who summarized only a single advanced query. We used ROUGE to compare the summaries and found significant difference in the ROUGE score - indicating that summaries produced at advanced stages of a chain are significantly affected by the context of the exploration session.

- **Can we use automatic summaries to improve the exploratory search process?**

  We did not address this question directly but we believe that our best preforming method (KLSum + LDA) can produce summaries that contain relevant information to the query and avoids redundancy between previous exploration steps. Those summaries can help the user make an informed decision on whether he wishes to further explore his current query or change the direction of the exploratory search.

6.2 Future Work

While creating and evaluating our automatically generated summaries, we found quantitative improvement but we also noticed three qualitative issues on which we should focus our future work:

6.2.1 Improving the coverage and redundancy elimination of our methods

All of our systems used bag of words (BOW) representation of sentences. Recent work [31] improved automatic summaries by applying anaphora resolution and replacing the anaphoric expressions with the object they are referring while creating BOW representation of sentences. For example the sentence 'Sally preferred the her own company' will be represented as $['sally','prefer','sally','company']$ instead of $['sally','prefer','her','company']$ and by that lexical analysis of the BOW model. We believe all of our methods could gain from applying this procedure.

6.2.2 Optimizing Runtime Performance

Our topic model requires re-assignment of all topics for each summary. If we could create a topic model that can capture relevancy and redundancy to an unknown query chain then we could run the model once and when the query is given, we could find the correct topic to capture new information. Such a model will greatly increase our run-time performance since the assignments of topics currently takes up to 30 minutes while the KLSum part of the method takes only few seconds for most queries.

6.2.3 Improving coherence

Most of the summaries generated by our top preforming system reported to be very informative but lacked coherence. Our basic sentence ordering scheme was not sufficient to achieve coherence. We believe that most our future work should

focus on improving coherence in summaries. We also believe that coherence cannot be achieved by extractive summarization even with manual sentence ordering different writing style between sentences taken from different documents do not achieve coherent summaries. We should try to explore abstractive summarizations schemes. With recent advances in the field of semantic parsing [32] and meaning representation [33] we believe we can capture logical representation of documents with variations of our methods to capture important information and then use text generation to achieve coherent summary.

Appendices

Appendix A

Query Chain Topic Model

In our work, we combine two LDA based summarization methods to achieve a query-chain system based on KLSum and topic modeling. The methods we used are TopicSum [28][1] and DualSum [35][2]. We present here the topic model we created to achieve this summarization method. We explain how by assigning specific topics to documents that are related to the current query or to the previous queries, we can assume those topics have a semantic meaning that we can use in a query-chain system.

Latent Dirichlet Allocation

Latent Dirichlet Allocation is a generative model that is commonly used for topic modeling. A topic model is a statistical model that maps words from a document set into a set of "abstract topics." The LDA model assumes that each document in the document set is generated as a mixture of topics with assigned probability to each topic. Once the topics of a document are assigned, words are sampled from

[1] A multi document summary method that uses LDA
[2] An update summary method that uses LDA

each topic to create the document. Given a set of documents, the LDA model tries to reproduce this latent process in order to uncover the topics that stand behind the words. Learning the probabilities of the topics is a problem of Bayesian inference. In the original LDA paper, Variational Bayes approximation is used to calculate the posterior distribution. Later work [36] used Gibbs sampling [37] to calculate the posterior distribution.

Gibbs sampling or a Gibbs sampler is a Markov Chain Monte Carlo (MCMC)[38] algorithm for obtaining a sequence of observations which are approximated from a specified multivariate probability distribution, when direct sampling is difficult. This sequence can be used to approximate the joint distribution, to approximate the marginal distribution of one of the variables, or some subset of the variables (for example, the unknown parameters or latent variables).

Our Topic Model

We identify a "new content" topic in a document collection by applying the following generative model:

- When given a query chain, we fetched the following documents set: all the documents that correspond with that previous queries in the chain D_p (top 10 TF/IDF scores for each query), the documents for the current query D_c (top 10 TF/IDF scores for the query), and all documents D.

- Our model generative story:

    1. G is the general words topic, it should capture stop words its distribution φ_G is drawn for all the documents from $Dirichlet(V, \lambda_G)$.

2. S_i is the document specific topic it should represent words which are local for a specific document ϕ_{Si} is drawn for each document from $Dirichlet(V, \lambda_{Si})$.

3. N is the new content topic that should capture words that are characteristic for D_c, ϕ_N is drawn for all the documents in D_c from $Dirichlet(V, \lambda_N)$.

4. O should capture old content from D_p, ϕ_O is drawn for all the documents in D_p from $Dirichlet(V, \lambda_O)$.

5. R topic should capture redundant information between D_c and D_p, ϕ_R is drawn for all the documents in $D_p \bigcup D_c$ from $Dirichlet(V, \lambda_R)$.

6. For each document from D_c we draw a distribution ψ_{t_1} over topics(G,N,R,S_i) from a Dirichlet prior with pseudo-counts[3] (10.0,15.0,15.0,1.0)[4] for each word in the document, we draw a topic Z from ψ_t, and a word W from the topic indicated by Z .

7. For documents from D_p we draw from the distribution ψ_{t_2} over topics(G,O,R,S_i) from a Dirichlet prior with pseudo-counts (10.0,15.0,15.0,1.0). The words were drawn in the same manner as in ψ_{t_1}.

8. For documents in $D \setminus (D_c \bigcup D_p)$ we draw from the distribution ψ_{t_3} over topics(G,S_i) from a Dirichlet prior with pseudo-counts (10.0,1.0). The words were also drawn in the same manner.

The Gibbs sampling equation were: $\theta_{sum_m,k} = \frac{nd_{m,k}+\alpha}{nd_sum_m+K*\alpha}$

$\phi_{sum_{k,w}} = \frac{nw_{w,k}+\text{getPseudoCount}(k)}{nw_sum_k+V*\text{getPseudoCount}(k)}$

Where θ_{sum} is cumulative statistics of theta, $nd_{m,k}$ is the number of words in doc-

[3]When applying Gibbs sampling we add the pseudo-counts to the actual count

[4]Values obtained empirically

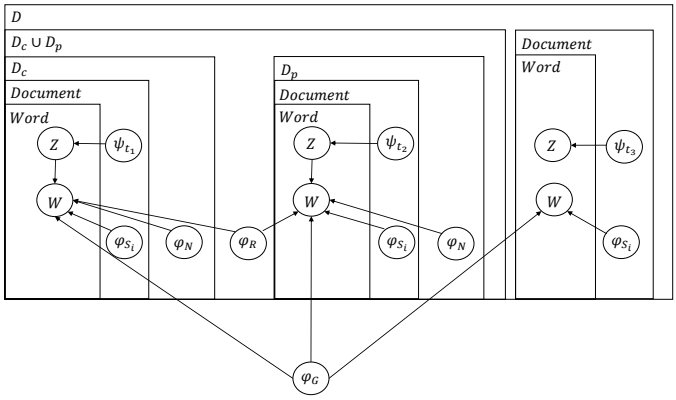

Figure A.1: Our topic model plate model

ument m assigned to topic k, nw_sum_k is the total number of words in document k, V is the vocabulary size, α is a Dirichlet parameter, $nw_{w,k}$ is the number of instances of word w assigned to topic k.

A.0.4 Evaluation

For evaluation of the topic models, we wrote down a few of the topic assignments of our topic model for the last query in the first chain *"asthma causes* $\rightarrow$ *asthma allergy* $\rightarrow$ **asthma mold allergy***"* to get a sense of the model performance.

- **General Topic** - 'will','call','doctor','test','type','physic','best','asthma'
- **New Content** - 'indoor','fungi','mold','microscop','grow'
- **Redundant content** - 'allerg','reaction','trigger','environmant','risk','problem'

From the assignment of topics we can notice that the general topic assigned with stop words not filtered in the tokenizing process such as 'will' and words

10_S $Ways_R$ to_X $Fight_G$ $Indoor_G$ $Mold_N$ $Mold_N$ is_X $among_X$ the_X $most_X$ $hazardous_N$ $household_N$ $substances_G$ for_X $people_G$ $with_X$ $allergies_R$ and_X $asthma_G$. It_X can_X $trigger_R$ $asthma_G$ $attacks_G$ and_X $allergic_R$ $reactions_S$ (such$_X$ as$_X$ hay$_N$ fever$_S$ or$_X$ eczema$_G$), and$_X$ it$_X$ can$_X$ even$_X$ cause$_X$ health$_G$ problems$_R$ in$_X$ people$_G$ without$_X$ those$_X$ conditions$_G$. $Mold_N$ $grows_R$ in_X $humid_N$, $damp_G$ $environments_N$, so_X the_X $best_R$ way_X to_X $prevent_G$ and_X $control_G$ $indoor_G$ $mold_N$ is_X to_X $keep_X$ $your_X$ $home_X$ as_X dry_R and_X $ventilated_R$ as_X $possible_X$. $While_X$ $bathrooms_S$ are_X an_X $obvious_S$ $place_G$ for_X $mold_N$ to_X get_X a_X $toehold_R$, $there_X$ are_X $some_X$ $other_X$ $relatively_X$ $surprising_S$ $sources_N$ of_X $mold_N$ in_X $your_X$ homesuch$_S$ as_X $firewood_S$.

G denotes word was assigned to the general topic.
N denotes word was assigned to the new content topic.
R denotes word was assigned to the redundant topic.
S denotes the word was assigned to the document specific topic.
X denotes the word was dropped in preprocessing[5] and was not assigned any topic.

Figure A.2: Query Chain Topic Model annotation of a document with current query 'asthma mold allergy' and previous queries are ['asthma causes','asthma allergy']

that are general to the asthma document set and general medicine words such as 'doctor' and 'asthma' together with common adjectives such as 'best'.

For the new content topic, the assignment is mainly related to 'mold' and for the redundant content topic most words are related to 'allergy' since both the last query and the 2nd query contained information about allergy.

Bibliography

[1] Robert S Taylor. The process of asking questions. *American Documentation*, 13(4):391–396, 1962.

[2] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[3] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

[4] Norbert Fuhr. Probabilistic models in information retrieval. *The Computer Journal*, 35(3):243–255, 1992.

[5] Meni Adler, Jonathan Berant, and Ido Dagan. Entailment-based text exploration with application to the health-care domain. In *Proceedings of the ACL 2012 System Demonstrations*, pages 79–84. Association for Computational Linguistics, 2012.

[6] Gary Marchionini. Exploratory search: from finding to understanding. *Communications of the ACM*, 49(4):41–46, 2006.

[7] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[8] Hoa Trang Dang and Karolina Owczarzak. Overview of the tac 2008 update summarization task. In *Proceedings of text analysis conference*, pages 1–16, 2008.

[9] Ani Nenkova and Rebecca J Passonneau. Evaluating content selection in summarization: The pyramid method. In *HLT-NAACL*, pages 145–152, 2004.

[10] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, 2004.

[11] Peter A Rankel, John M Conroy, Hoa Trang Dang, and Ani Nenkova. A decade of automatic content evaluation of news summaries: Reassessing the state of the art. 2013.

[12] Jonathan Berant, Ido Dagan, and Jacob Goldberger. Global learning of focused entailment graphs. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1220–1229. Association for Computational Linguistics, 2010.

[13] Raphael Cohen, Michael Elhadad, and Ohad Birk. Analysis of free online physician advice services. *PloS one*, 8(3):e59963, 2013.

[14] Günes Erkan and Dragomir R Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Res.(JAIR)*, 22(1):457–479, 2004.

[15] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[16] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1):107–117, 1998.

[17] Fan Chung and Wenbo Zhao. Pagerank and random walks on graphs. In *Fete of Combinatorics and Computer Science*, pages 43–62. Springer, 2010.

[18] Yajie Miao and Chunping Li. Enhancing query-oriented summarization based on sentence wikification. In *Workshop of the 33 rd Annual International*, page 32, 2010.

[19] Lev-Arie Ratinov, Dan Roth, Doug Downey, and Mike Anderson. Local and global algorithms for disambiguation to wikipedia. In *ACL*, volume 11, pages 1375–1384, 2011.

[20] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl 1):D267–D270, 2004.

[21] S Lipsky-Gorman and N Elhadad. Clinnote and healthtermfinder: a pipeline for processing clinical notes. *Columbia University*, 2011.

[22] David Milne. Computing semantic relatedness using wikipedia link structure. In *Proceedings of the new zealand computer science research student conference*. Citeseer, 2007.

[23] Bridget T McInnes, Ted Pedersen, and Serguei VS Pakhomov. Umls-interface and umls-similarity: open source software for measuring paths and semantic similarity. In *AMIA Annual Symposium Proceedings*, volume 2009, page 431. American Medical Informatics Association, 2009.

[24] Baoli Li, Joseph Irwin, Ernest V Garcia, and Ashwin Ram. Machine learning based semantic inference: Experiments and observations at rte-3. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 159–164. Association for Computational Linguistics, 2007.

[25] Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543. ACM, 2002.

[26] Yllias Chali and Shafiq R Joty. Improving the performance of the random walk model for answering complex questions. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 9–12. Association for Computational Linguistics, 2008.

[27] Sampo Pyysalo, Tapio Salakoski, Sophie Aubin, and Adeline Nazarenko. Lexical adaptation of link grammar to the biomedical sublanguage: a comparative evaluation of three approaches. *BMC bioinformatics*, 7(Suppl 3):S2, 2006.

[28] Aria Haghighi and Lucy Vanderwende. Exploring content models for multi-document summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the*

Association for Computational Linguistics, pages 362–370. Association for Computational Linguistics, 2009.

[29] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[30] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[31] Daniel Alexandru Anechitei and Eugen Ignat. Multilingual summarization system based on analyzing the discourse structure at multiling 2013. *MultiLing 2013*, page 72, 2013.

[32] Yoav Artzi, Nicholas FitzGerald, and Luke Zettlemoyer. Semantic parsing with combinatory categorial grammars.

[33] Cai Banarescu, Bonial. Abstract meaning representation @ONLINE, 2013.

[34] Jean-Yves Delort and Enrique Alfonseca. Dualsum: a topic-model based approach for update summarization. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 214–223. Association for Computational Linguistics, 2012.

[35] Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235, 2004.

[36] Chang-Jin Kim and Charles R Nelson. State-space models with regime switching: classical and gibbs-sampling approaches with applications. *MIT Press Books*, 1, 1999.

[37] Arnaud Doucet, Nando De Freitas, Neil Gordon, et al. *Sequential Monte Carlo methods in practice*, volume 1. Springer New York, 2001.