Sherlock: A System for Interactive Summarization of Large Text Collections

Avinesh P.V.S.[1], Benjamin Hättasch[1], Orkan Özyurt[1],
Carsten Binnig[1,2], Christian M. Meyer[1]
[1] TU Darmstadt, Germany [2] Brown University, USA
{avinesh,oezyurt,meyer}@ukp.tu-darmstadt.de,{benjamin.haettasch,carsten.binnig}@cs.tu-darmstadt.de

ABSTRACT

There exists an ever-growing set of data-centric systems that allow data scientists of varying skill levels to interactively manipulate, analyze and explore large structured data sets. However, there are currently not many systems that allow data scientists and novice users to interactively explore large unstructured text document collections from heterogeneous sources.

In this demo paper, we present a new system for interactive text summarization called Sherlock. The task of automatically producing textual summaries is an important step to understand a collection of multiple topic-related documents. It has many real-world applications in journalism, medicine, and many more. However, none of the existing summarization systems allow users to provide feedback at interactive speed. We therefore integrate a new approximate summarization model into Sherlock that can guarantee interactive speeds even for large text collections to keep the user engaged in the process.

PVLDB Reference Format:
Avinesh P.V.S., Benjamin Hättasch, Orkan Özyurt, Carsten Binnig, and Christian M. Meyer. Sherlock: A System for Interactive Summarization of Large Text Collections. *PVLDB*, 11 (12): 1902–1905, 2018.
DOI: https://doi.org/10.14778/3229863.3236220

1. INTRODUCTION

Motivation: Existing data-centric systems for interactively manipulating, analyzing and exploring large data sets focus particularly on structured data. For example, tools like Tableau or Cognos allow to quickly analyze high-dimensional data using an interactive and visual interface. Research prototypes like imMens [13], DICE [8, 9] or IDEA [3] aim to improve upon systems like Tableau by using specialized data structures, pre-fetching and/or approximation techniques to guarantee interactive latencies over big data sets. Other research projects like SeeDB [11] or Data Polygamy [2], help users during the exploration process by providing recommendations for interesting visualizations or correlations, whereas systems like DataWrangler [10], Trifacta [17] or Paxata [16] assist users in data wrangling and cleaning.

Proceedings of the VLDB Endowment, Vol. 11, No. 12
Copyright 2018 VLDB Endowment 2150-8097/18/8.
DOI: https://doi.org/10.14778/3229863.3236220

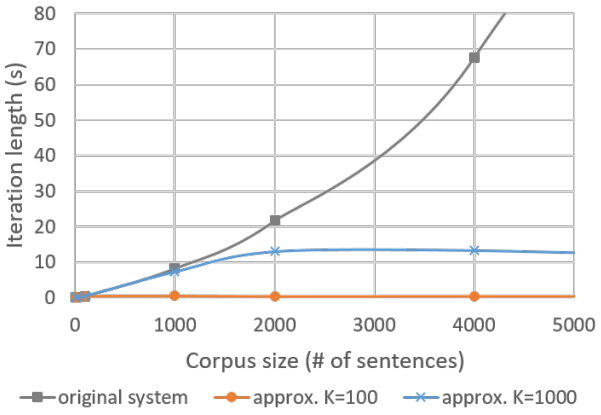

Figure 1: Scalability of Text Summarization Models

While the before-mentioned interactive exploration systems target mainly structured data, there are currently not many systems apart from [6] and [5] that allow data scientists of varying skill levels and novice users to interactively explore unstructured text document collections. Most of the existing interactive text exploration systems, however, only allow users to apply classical keyword searches [6] and rank the documents according to static metrics (e.g., frequency [1] or centrality [15]). While keyword-based search systems are important to filter down the number of relevant documents, they still do not support users in understanding the document collection. Imagine for example a journalist who just received a large collection of documents to start an investigative case, or a lawyer who needs to screen a large collection of e-mail conversations. In all these examples, an important step to better understand the collection of documents is to produce a concise textual summary that captures most of the important information relevant to a user's individual goal.

The task of producing textual summaries from a collection of multiple topic-related documents is a well-established task in the text analysis community. Despite a lot of research in this area, it is still a major challenge to automatically produce summaries that are on par with human-written ones. To a large extent, this is due to the complexity of the task: a good summary must include the most relevant information, omit redundancy and irrelevant information, satisfy a length constraint, and be cohesive and grammatical. But an even bigger challenge is the high degree of subjectivity in the summarization task, as it can be seen in the small overlap of what is considered important by different users [14]. Optimizing a system towards one single best summary that fits all users, as it is

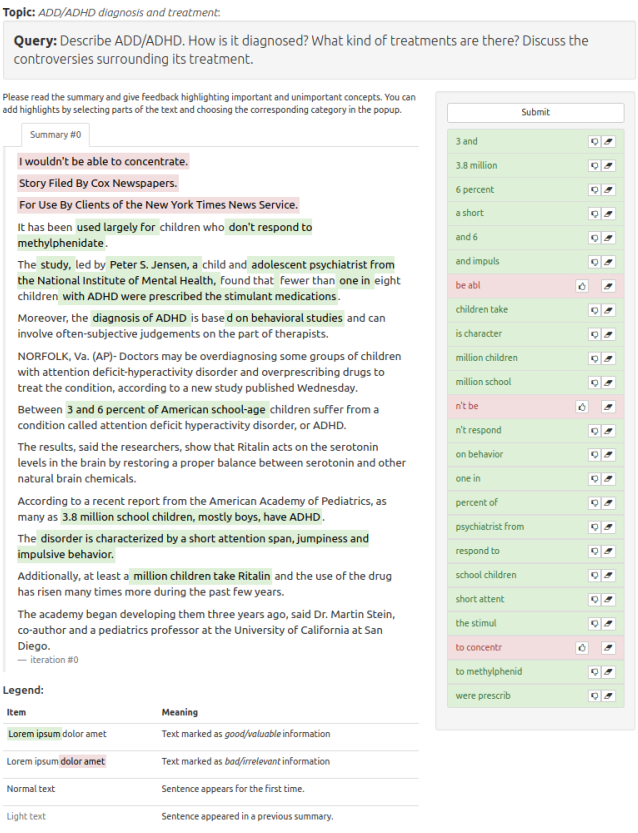

Figure 2: A Screenshot of Sherlock

assumed by current state-of-the-art systems, is highly impractical and diminishes the usefulness of a system for real-world use cases.

Contributions: To this end, in a recent research paper [14] we have suggested an interactive concept-based model to assist users in creating a personalized summary based on their feedback. While we could show that user feedback significantly improves the quality of the summary, each iteration in our model can take from several seconds for small document collections to hours for larger collections with thousands of sentences as shown in Figure 1. However, one of the most important aspects is to keep users involved in the exploration process. In fact, a recent study [12] has shown that even small delays (more than 500 ms) significantly decrease a user's activity level, dataset coverage, and insight discovery rate.

In this demo paper, we build on our own prior work [14] and integrate the interactive summarization model into a new system with an interactive user interface called Sherlock. A screenshot of the web-based user interface of our system can be seen in Figure 2. Using our system, in every interaction loop the system suggests a potential summary and the user can accept or reject individual concepts (e.g., entities) shown as green or red in the suggested summary. The feedback is then used to refine the summary for the next iteration. In order to get a better overview of how the system works, we recommend the readers to watch the demo video[1].

In order to provide interactive response times, the backend of Sherlock implements a novel approximate version of our interactive summarization model. The main idea is similar to sample-based approximate query processing in databases: Instead of looking at the complete document collection in every iteration, we only consider a sample from the documents per iteration to compute the summary in a given time threshold (e.g., 500 ms). We therefore developed a new sampling strategy to select a smaller set of "promising" sentences from all input documents. As we show in Figure 3, that way our approximate summarization model can provide interactive latency for each interaction loop independent of the size of the text collection that is being summarized.

[1]http://vimeo.com/257601765

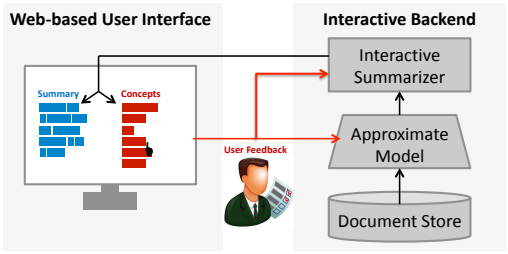

Figure 3: Sherlock System Overview

Outline: The remainder of the demo paper is structured as follows: In Section 2 we give an architectural overview of our system and describe the high-level idea of the interactive summarization procedure. We then discuss the details of the existing interactive summarization model and our new extensions to support approximate summaries in Section 3. Afterwards, we discuss the demo scenario that shows how users can interact with our system to produce text summaries interactively for document collections of different sizes.

2. SYSTEM OVERVIEW

The Sherlock system consists of two major components as shown in Figure 3: a web-based user interface and a novel backend that implements our interactive text summarization model. The backend hosts multiple components, a document store as well as the required indexes, the summarization component that accumulates the user feedback and creates the summaries for every iteration as well as an approximation model to enable an interactive execution of the summarization.

User Interface: We designed a web-based interface that allows users to summarize a collection of textual documents in an interactive manner. A screenshot was already shown in Figure 2. In a typical setup, a user would need to read all the documents and manually summarize it. In our interactive approach, the user is shown a summary and marks all the important concepts (marked green in Figure 2) and the unimportant concepts (marked red in Figure 2). The accepted and rejected concepts then appear on the right-hand side of the user interface. In the final step, the user can review his feedback and submit them for the next iteration.

Interactive Backend: The main task of the backend is to compute the text summary for each iteration by taking the documents and the user feedback into account. The first summary which is presented to the user is created without any feedback. Afterwards, in every iteration the summarization component takes the user feedback of all previous iterations into account. The current version of the backend implements both summarization models; the original one as well as the new approximate one that relies on sampling. An important component that is leveraged by the the new approximate summarization model is the cost-based approximation model

which selects only a relevant sample of sentences for the summarization component. An important task of the approximation model is to estimate the size of the sample for a given document collection and an interactivity threshold (e.g., 500ms) that should be met by the backend. The details of our summarization model are explained in the next section.

3. INTERACTIVE SUMMARIZATION

3.1 Basic Summarization Model

ILP-based summarization: Boudin et al. [1] propose an automatic summarization framework based on integer linear programming (ILP), whose goal is to select sentences $S_t \subseteq D$ from a document collection D that maximize the occurrence of important concepts $c \in C$ (e.g., bigrams, named entities, syntactic phrases) in S_t under a given length constraint L for S_t. Binary variables $c_i, i \in C$ and $s_j, j \in D$ of the ILP indicate if a concept ($c_i = 1$) or sentence ($s_j = 1$) is present in the resulting summary S_t. The ILP is formulated as follows:

$$\max \quad \sum_{i=1}^{|C|} w_i c_i \tag{1}$$

$$\sum_{j=0}^{|D|} \ell_j s_j \leq L \tag{2}$$

$$\forall i \in C, j \in D. \quad s_j \cdot Occ_{ij} \leq c_i \tag{3}$$

$$\forall i \in C. \quad c_i \leq \sum_{j=1}^{|D|} Occ_{ij} \cdot s_i \tag{4}$$

$$\forall i \in C. \quad c_i \in \{0, 1\} \tag{5}$$

$$\forall j \in D. \quad s_j \in \{0, 1\} \tag{6}$$

Equation (1) is the optimization goal of the ILP: The concepts covered by S_t should yield a maximal overall weight. As concept weights w_i, we use the document frequency (i.e., number of documents in D that contain concept i). The length of S_t is the sum of all sentence lengths ℓ_j of all sentences $j \in S_t$, which is constrained by the length parameter L (2). The equations (3) and (4) ensure the selection of all concepts in a selected sentence j and that a concept i is only selected if it is present in at least one selected sentence. The last equations (5) and (6) constrain c_i and s_j to binary values.

Algorithm 1 Basic Interactive Summarization

```
1: procedure InteractiveSummarizer()
2:     input: Documents D
3:     C ← extractConcepts(D)
4:     W ← conceptWeights(C)
5:     for t = 0...T do
6:         S_t ← getSummary(D, C, W)
7:         C_t ← extractConcepts(S_t)
8:         if ⋃_{τ=0}^{t} C_τ = C then
9:             return S_t
10:        else
11:            R_t ← obtainFeedback(S_t, C_t)
12:            W ← updateWeights(W, R_t, C_t)
13:        end if
14:    end for
15: end procedure
```

Interactive summarization: On the basis of this model, we recently proposed an interactive summarization approach [14] that learns from user feedback. Algorithm 1 shows this procedure in pseudo code. Given a document collection D, we first extract the concepts C and initialize the set of concept weights W (lines 3–4). In an interactive loop over $t = 0, \ldots, T$ (line 5), we collect feedback from the user. In every iteration, a summary S_t (line 6) is created using the concept-based ILP summarization framework discussed above. We display this summary along with its concepts C_t in our web-based UI. Afterwards, the user may accept concepts considered important or reject irrelevant concepts by clicking. Based in this feedback R_t, we update the concept weights by increasing the weight of important concepts and decreasing the weights for irrelevant ones (see [14] for details). We terminate the summarization after T iterations, if the user is satisfied with the result, or if all concepts have been seen. For small reference datasets from the summarization community, ten iterations are typically sufficient when assuming perfect feedback [14].

3.2 Approximate Summarization Model

The main problem of the basic summarization model discussed before is that the runtime to provide a summary in every iteration ranges from seconds for small collections of ten documents up to hours for larger collections of several hundred documents. However, when users are involved they want to provide feedback in an interactive manner instead of waiting potentially for hours to start the next feedback loop. We therefore extended Algorithm 1 as follows: Instead of executing the ILP-based summarization model on all sentences D of the document collection, we only use a sample D_t of sentences in every iteration t. In our extended version of the algorithm, the sample D_t is used as the input to create the summary (line 6) instead of all sentences.

For creating the sample D_t, two important factors play a role: The first factor is the sample size $K = |D_t|$ (i.e., the number of sentences in the sample), which determines the runtime of the summarization method. The second factor is the sampling procedure, that determines *which* K sentences are part of the sample. In the following, we discuss how these two factors are implemented in our current prototype of Sherlock.

In order to determine the sample size K, we need to define an interactivity threshold (say 500ms [12]) that an iteration should take maximally. Based on this threshold, we then derive a maximal K such that the resulting ILP can be solved in the given threshold using a cost model. Therefore, the cost model first needs to determine the approximate complexity of the ILP (in number of constraints) for a given K and then needs to estimate the runtime of solving this ILP. For estimating the runtime, we rely on calibration runs with the particular ILP solver that should be used in Sherlock to map a given number of constraints to an estimated runtime.

For deciding which sentence should be contained in the sample D_t, we developed a novel heuristic called information density that is computed for each sentence in D. In a nutshell, the heuristic is the weight density of concepts normalized by the sentence length. For sampling, we then only select the top-K sentences based on this heuristic. The intuition is that sentences with a higher information density are more relevant to the user. It is important to note, that the information density of a sentence changes based on the user feedback since the feedback is used to update the weights of concepts and this also changes the information density of sentences that contain those concepts. For example, if all concepts of a sentence are not seen to be relevant based on the feedback, their weights will all be set to 0 and thus the information density of that sentence will also be 0.

4. DEMONSTRATION

In our demonstration video at https://vimeo.com/257601765, we show two scenarios of how Sherlock[2] can be used interactively to

[2]https://sherlock.ukp.informatik.tu-darmstadt.de

produce personalized summaries. We describe these two scenarios below.

Query-driven summarization: As our first scenario, consider parents of an elementary student. As they are worried about their kid having ADHD, they explore a topically focused document collection in order to learn, how ADHD is diagnosed and treated. In this scenario, we use a query-focused summarization corpus DUC '06 [4]. The parents start by exploring a preliminary summary of the document collection by entering their feedback on important concepts such as 'behavioral studies', 'short attention span', 'jumpiness', 'impulsive behavior', 'stimulant medication', 'Ritalin acts on the serotonin levels in the brain', 'children who do not respond to methylphenidate', etc. They also reject unrelated concepts like 'overprescribing drugs', 'academy began developing', and 'doctors may be overdiagnosing'. The parents then review the accepted and the rejected concepts on the right-hand side of the user interface and submit them to let the system improve the summary of the next iteration. Sherlock learns from this feedback and creates a new summary based on the adapted concept weights. This process continues for five to six iterations, until the parents fulfilled their information need and they are satisfied with the resulting summary text.

Exploratory summarization: As the document collection used grows larger, the efficiency of the approach becomes a severe bottleneck. Runtimes of tens of seconds per iteration and more reduce the user's activity level and thus limit the effectiveness of the system. Sherlock's approximate model prevents this even for large document collections of thousands of sentences. Our second scenario particularly highlights the need for efficient system responses: Imagine a journalist investigating the situation in schools. This scenario deals with the exploration of large document collection (more than 1,000 documents) for a user's information need using the DIP corpus [7]. As her exact information need is yet unclear, she explores a large collection of educational reports, web documents, and forum entries using Sherlock. During the first iterations, the summary is yet very broad and generic, but highlights some of the issues discussed in the documents. The journalist rejects concepts that do not seem interesting for her news story, such as 'legal issues' or 'curriculum'. She, however, also identifies controversial issues about 'handling conflicts between children', 'bullying', or 'religious classes'. By exploring the document collection further, spanning multiple iterations, the journalist decides to write a news story about parents' concerns about religious classes in school, starting from Sherlock's personalized summary, that she revises and extends by additional facts and interviews.

5. CONCLUSION AND FUTURE WORK

In this demo paper, we presented a new system for interactive text exploration and summarization at interactive speed. We propose an approximate model that bounds the runtime to engage the user's activity. We discuss two usage scenarios illustrated in a video. Our system is however applicable to a wide range of realistic use cases, including the interactive summarization of legal documents for a particular case or exploring medical publications for adverse drug reaction.

6. ACKNOWLEDGMENTS

This work has been supported by the German Research Foundation as part of the Research Training Group "Adaptive Preparation of Information from Heterogeneous Sources" (AIPHES) under grant No. GRK 1994/1. We thank Hendrik Lücke-Tieke for his assistance with the GUI.

7. REFERENCES

[1] F. Boudin, H. Mougard, and B. Favre. Concept-based summarization using integer linear programming: From concept pruning to multiple optimal solutions. In *EMNLP*, pages 1914–1918. ACL, 2015.

[2] F. Chirigati, H. Doraiswamy, T. Damoulas, and J. Freire. Data polygamy: the many-many relationships among urban spatio-temporal data sets. In *ACM SIGMOD*, pages 1–15. ACM, 2016.

[3] A. Crotty, A. Galakatos, E. Zgraggen, C. Binnig, and T. Kraska. The case for interactive data exploration accelerators (IDEAs). In *HILDA@SIGMOD*, page 11. ACM, 2016.

[4] Document understanding conference 2006 corpus. http://duc.nist.gov/duc2006. Accessed: 2018-03-01.

[5] T. Falke and I. Gurevych. GraphDocExplore: A framework for the experimental comparison of graph-based document exploration techniques. In *EMNLP*, pages 19–24. ACL, 2017.

[6] D. Glowacka, T. Ruotsalo, K. Konuyshkova, K. Athukorala, K. Samuel, and G. Jacucci. Directing exploratory search: Reinforcement learning from user interactions with keywords. In *ACM IUI*, pages 117–127. ACM, 2013.

[7] I. Habernal, M. Sukhareva, F. Raiber, A. Shtok, O. Kurland, H. Ronen, J. Bar-Ilan, and I. Gurevych. New Collection Announcement: Focused Retrieval Over the Web. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 701–704, 2016.

[8] P. Jayachandran, K. Tunga, N. Kamat, and A. Nandi. Combining user interaction, speculative query execution and sampling in the dice system. *PVLDB*, 7(13):1697–1700, 2014.

[9] N. Kamat, P. Jayachandran, K. Tunga, and A. Nandi. Distributed and interactive cube exploration. In *ICDE*, pages 472–483. IEEE, 2014.

[10] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. Wrangler: Interactive visual specification of data transformation scripts. In *ACM SIGCHI*, pages 3363–3372. ACM, 2011.

[11] M.-T. Ke, S. Fujimoto, and T. Imai. Seedb: a simple and morphology-preserving optical clearing agent for neuronal circuit reconstruction. *Nature neuroscience*, 16:1154–1161, 2013.

[12] Z. Liu and J. Heer. The effects of interactive latency on exploratory visual analysis. *IEEE transactions on visualization and computer graphics*, 20:2122–2131, 2014.

[13] Z. Liu, B. Jiang, and J. Heer. immens: Real-time visual querying of big data. In *Computer Graphics Forum*, volume 32, pages 421–430. Wiley Online Library, 2013.

[14] Avinesh P. V. S. and C. M. Meyer. Joint optimization of user-desired content in multi-document summaries by learning from user feedback. In *ACL*, pages 1353–1363. ACL, 2017.

[15] R. Mihalcea and P. Tarau. Textrank: Bringing order into texts. In *EMNLP*, pages 404–411. ACL, 2004.

[16] Paxata. http://www.paxata.com, 2018. Accessed: 2018-03-01.

[17] Trifacta. http://www.trifacta.com, 2018. Accessed: 2018-03-01.