Interactive Summarization of Large Document Collections

Benjamin Hättasch
TU Darmstadt

Christian M. Meyer
TU Darmstadt

Carsten Binnig
TU Darmstadt

ABSTRACT

We present a new system for custom summarizations of large text corpora at interactive speed. The task of producing textual summaries is an important step to understand large collections of topic-related documents and has many real-world applications in journalism, medicine, and many more. Key to our system is that the summarization model is refined by user feedback and called multiple times to improve the quality of the summaries iteratively. To that end, the human is brought into the loop to gather feedback in every iteration about which aspects of the intermediate summaries satisfy their individual information needs. Our system consists of a sampling component and a learned model to produce a textual summary. As we show in our evaluation, our system can provide a similar quality level as existing summarization models that are working on the full corpus and hence cannot provide interactive speeds.

1 INTRODUCTION

Motivation: Existing data-centric systems for interactively manipulating, analyzing and exploring large data sets focus particularly on structured data. However, in many use cases the relevant data sources are not structured but are only present as a collection of texts. There already exist systems for text exploration like [3] and [2] that allow data scientists of varying skill levels and novice users to interactively analyze unstructured text document collections—however, those systems concentrate mainly on keyword searches and document ranking.

While keyword-based search systems are important to filter down the number of relevant documents, they still do not support users in semantically 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 text and find the overall relation and event structure of those 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 documents is a well-established task in the text analysis community [7]. 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. 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 [6]. Optimizing a system towards one single best summary that fits all users, as it is assumed by current state-of-the-art systems, is highly impractical and diminishes the usefulness of a system for real-world use cases.

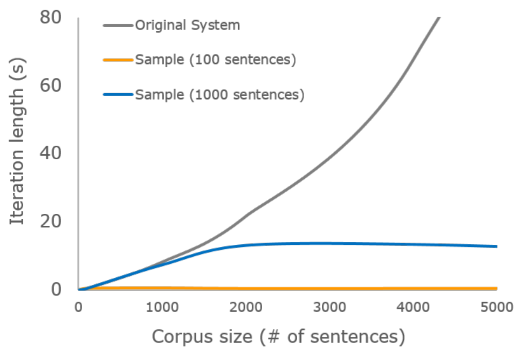

Figure 1: Scalability of Text Summarization Models [5]

In a recent paper [6], we have shown that user feedback significantly improves the quality of the summary. However, each iteration of learning to create a new summary based on the user's feedback can take from several seconds for small document collections to hours for larger collections as shown in Figure 1 (black line). Since the customization of the summary depends on the user's feedback, it is one of the most important aspects to keep users involved in the exploration process. Yet this can hardly be reached with long iteration times of multiple hours, even waiting times of minutes or multiple seconds can already cause the user to loose interest. A previous study [4] has shown that even small delays of more than 500ms significantly decrease a user's activity level, dataset coverage, and insight discovery rate.

Contributions: In this paper, we present Sherlock that allows users to interactively summarize large text collections. In order to provide interactive response times in each iteration of the summarization procedure, we are using a novel approximate summarization model. The main idea of the approximate summarization model is similar to 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. As a main contribution, we propose a method to select the sample size based on iteration time thresholds and evaluate multiple different sampling strategies. As we show in Figure 1, 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 (orange and blue line).

HILDA'19, July 5, 2019, Amsterdam, Netherlands
© 2019 Copyright held by the owner/author(s). Publication rights licensed to Association for Computing Machinery.
ACM ISBN 978-1-4503-6791-2/19/07…$15.00
https://doi.org/10.1145/3328519.3329129

We already presented a demo of Sherlock at VLDB 2018 [5]. However, sampling on natural language is not a trivial task. In this paper, our main goal is studying the effectiveness of multiple sampling strategies and the impact of the sample size on the summarization quality. To this end, we employ importance-based and stratified sampling, and we benchmark them in a systematic experimental setup on different document collections.

A related paper was recently published for sampling training data for machine learning [8]. In their paper, the authors suggest to use controlled sampling for architecture selection to predict the errors introduced by the approximate model. While their work concentrated mainly on classical models learned from structured data, we will investigate the effects of sampling for textual summarization models with different desired properties (e.g., subjective importance of concepts or wide vs. focused textual summaries).

Outline: The remainder of the paper is structured as follows: In Section 2 we describe the high-level idea of our system for interactive summarization and then propose different sampling strategies and explain the sample size estimation in Section 3. We then show an initial experimental evaluation of our novel sampling strategies on the model quality in Section 4.

2 SYSTEM OVERVIEW

Sherlock [5], our system for interactive summarization, consists of two major components as shown in Figure 2: a web-based user interface to collect the user's feedback and a backend that refines the text summarization model. The backend hosts multiple components: a document store (input docs in Figure 2) including the required indexes, the summarization component that accumulates the user feedback and learns to create summaries for every iteration as well as our approximate model to execute the summarization process at interactive speeds.

User Interface: The web-based interface allows users to summarize a collection of textual documents in an interactive manner. A screenshot of the interface is included in Figure 2. In a typical setup, a user would need to read all the documents and manually summarize them. In our interactive setup, the user receives a summary, annotates all important and unimportant (parts of) sentences, and submits them as feedback for the next iteration where a refined summary is created by Sherlock and the user provides the next round of feedback.

Interactive Backend: The main task of the backend is to compute the summary for each iteration by taking the documents and the user feedback into account. In our system, we currently employ the summarization model as presented in [6] which maximizes the weighted occurrence of concepts in the summary by using an integer linear program (ILP). The first summary which is presented to the user is based on a model without any user feedback. Afterwards, in every iteration the summarization model is refined based on the user feedback of all previous iterations; i.e., the user can adjust the concept weights and hence the ILP needs to be re-executed. Instead of using the full document corpus as input, our backend uses samples of a given size (from very small to full data) resulting in different iteration times and expected quality. The details of our approximate summarization model are explained in the next section.

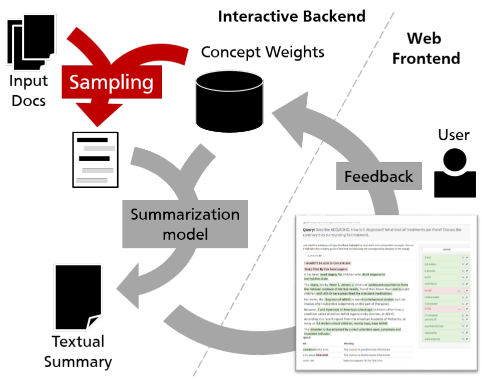

Figure 2: System Overview of Sherlock.

3 APPROXIMATE SUMMARIZATION MODEL

The main idea of our approximate summarization model is to take the user feedback of the last iteration into account, adjust the summarization model and then return a new version of the summary to the user. As discussed before, in order to achieve interactive response times in every iteration, the approximate summarization model takes a sample of the overall document collection as input. The sampling strategy and the sample size have a big impact on the performance and the quality of the summarization model. In the following subsections, we discuss both these aspects in detail.

3.1 Sampling Strategies

In this first subsection, we discuss which sentences should be sampled in every iteration to refine the trained summarization model. There are numerous possibilities. In this paper, we suggest and evaluate three sampling strategies tailored towards training textual summarization models:

Random: This is the simplest sampling strategy where we just use a fixed number of randomly selected sentences in each iteration. This method is used as a baseline compared to the more sophisticated sampling strategies discussed next.

Importance-based (called TOP-K as well): Instead of sampling randomly, we suggest a second strategy that takes the importance of a sentence into account when sampling from the underlying document collection (i.e., more important sentences are sampled with a higher likelihood). Our intuition is that sentences with a higher *information density* (containing more concepts rated as important) are more relevant to the user. As concepts, we use bigrams in our system as suggested by [6]. We initialize the weight of a concept using the *document frequency*; i.e., the number of documents in the collection the concept appears in. The information density of a sentence is the average weight of all concepts in the sentence. Based on the user feedback, we increase and decrease the weights of the concepts yielding refined information density scores. In every iteration, we induce a sentence ranking and select only the top-*k* sentences based on the information density. This strategy introduces some computation overhead but allows exploitation of collected feedback already in the sampling process.

Stratified: In the third sampling strategy, we additionally divide the input sentences into a fixed number of clusters based on sentence embeddings [1]. Each of those clusters is individually ranked as discussed before. Based on the feedback, more sentences from

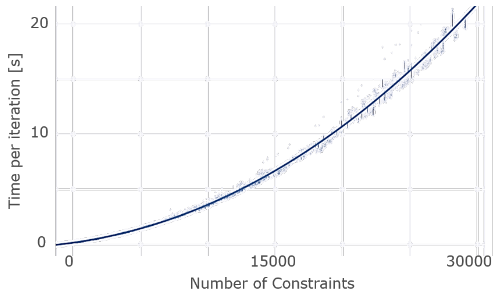

Figure 3: Number of constraints in ILP in relation to estimated / actual runtime for finding the best summary (in seconds).

clusters with better feedback are sampled. This allows dealing with diverse topics and causes the sampling to initially better explore all information available (instead of only the frequent concepts).

3.2 Sample Size Estimation

For all the before-mentioned sampling strategies one needs to choose the sample size as a parameter. This parameter should not be selected arbitrarily since a too small or too large sample size might have a negative impact on the overall quality—by not sampling relevant sentences as well as by changing the runtime of the summarization procedure which is important to enable the user to give interactive feedback.

At the core of our system, as discussed before, an ILP solver is used that maximizes the accumulated weight of all concepts in the summary for a given summary length (i.e. consists of sentences mainly containing the highest-rated distinct concepts based on the user feedback). In order to set the sample size, we use a *cost model* to estimate the response time of the system. In future, we will extend this cost model to enable us to estimate the resulting quality of the summarization model when using only a sample of a given size.

The main intuition behind our cost model is that each sentence in the input to the ILP produces additional constraints that have to be respected for finding the summary in the next iteration by the solver. Less constraints make it easier for the solver to find a solution and therefore reduce the computation time. The cost function thus only depends on the sample size (which directly translates into the number of constraints) but not the summary length, since this is only present as a single constraint in the ILP. Hence, different summary lengths with the same amount of input concepts will still yield similar runtimes of the summarization system. We verify those findings at the beginning of our evaluation (Section 4.1).

4 EXPERIMENTAL EVALUATION

4.1 Exp. 1: Sample Size

In a first experiment, we analyze the accuracy of our cost model to estimate the sample size. As discussed before, the cost function in Sherlock maps the number of constraints to an estimated runtime. We use this function to derive the maximum sample size k such that the runtime stays below a chosen interactivity threshold (e.g., 500 ms). Figure 3 compares the actual runtime of the ILP model with the estimated runtime of our cost model (blue line). Both show a Pearson correlation coefficient of 0.96 to 0.97.

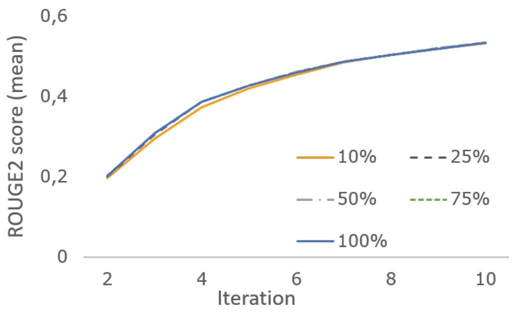

Figure 4: Mean recall of bigrams (ROUGE2 metric) for running the system with 5 different sample sizes (from 10% to full data) on 615 different data sets (artificial and real data). All experiments use the TOP-K strategy.

Furthermore, another important question is the effect of using a sample as input to the summarization model. Figure 4 shows the results of running the summarization procedure on different sample sizes using the TOP-K strategy (while STRATIFIED behaved similar in this experiment) over the different iterations (shown on the x-axis) to collect feedback from users. For collecting feedback, we simulate users in all our experiments who give perfect feedback to produce summaries (i.e., users always mark concepts as important if they appear in the gold summary). As a result, we see that the quality of summaries produced with only 10% sample size is nearly the same as for the full data and the mean quality of the system working on a quarter of the input data is indistinguishable from the mean quality of the original system while the length of each iteration is only about 20% of the one from the original system.

Another interesting observation is that the sample cannot be arbitrarily small—it still has to contain enough sentences to fill the full summary and the summarizing model should still be able to choose from different sentences. In our current prototype, we thus select the sample size based on our cost model such that the desired interaction threshold is met but the sample size still contains enough sentences to fill the full summary (i.e., sample size must be larger than the summary length).

4.2 Exp. 2: Sampling Strategy

In the second experiment, we evaluate the different sampling strategies and their robustness for different kinds of summaries. There are two important dimensions of textual summaries that have a major impact on how well a sampling strategy works: The first dimension is which concepts are included in a summary (i.e., frequent ones or rare ones). The second dimension is whether the summary is topically focused (for users who are interested in particular details) or if it contains a wide range of concepts (for users who want to get an initial overview).

In order to evaluate our sampling strategies on these different summary types, we create different summaries (called gold summaries) that follow the above-mentioned properties. To control the content of the summaries and make sure they still have the syntactic properties and word distributions of a real text, we use sentences from the existing summarization corpora DUC06 and DUC07 to create the gold summaries.[1]

[1]https://www-nlpir.nist.gov/projects/duc/data.html

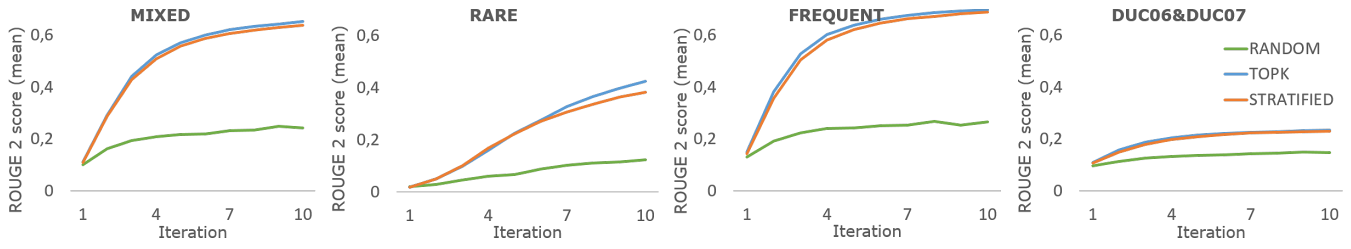

Figure 5: Mean ROUGE2 scores for running the system for 10 iterations with three different strategies (RANDOM, TOP-K, STRATIFIED) on our summaries (MIXED, RARE and FREQUENT from left to right) as well as real data (rightmost plot, original DUC06 & DUC07). 380 different summaries were used per plot. ROUGE2 scores are always between 0 and 1, higher scores are better.

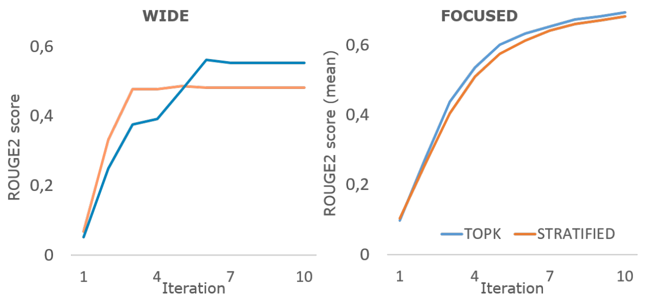

Figure 6: ROUGE2 scores for running the system for 10 iterations on a WIDE summary (left) and mean ROUGE2 scores for 50 artificial datasets with FOCUSED summaries (right).

Below, we first describe the setup for two experiments we conducted and then discuss the results at the end. We fix the sample size to 20% of the input documents for all different sampling strategies (RANDOM, TOP-K, STRATIFIED).

Exp. 2a: Rare vs. Frequent: In this experiment, we test the dependence of the sampling strategies on the frequency of the concepts. We therefore create three different classes of summaries: (1) summaries with random concepts containing both rare and frequent concepts (called MIXED), (2) summaries with concepts that appear frequently (called FREQUENT), and summaries with only concepts that are rare (called RARE). Furthermore, we use the summaries included in the original DUC corpora.

Exp. 2b: Focused vs. Wide: In this experiment, we want to test the sampling strategies on summaries with a rather focused or a wide set of topics. In order to create those summaries, we use the GloVe word embeddings [9] and sample sentences where words are uniformly distributed over the vector space (called WIDE) or sentences where words are clustered in a certain region (called FOCUSED).
All gold summaries have a length of about 250 words.

Discussion of Results: The results of Exp. 2a (Rare vs. Frequent) are shown in Figure 5. Again, as before we show the ROUGE2 metric over the different iterations of producing a summary whereas users provide perfect feedback in each iteration. For the result in Figure 5, we see that TOP-K and STRATIFIED sampling clearly outperform the RANDOM sampling strategy and thus both can be used to generate high-quality summaries in only a few iterations. Moreover, as expected both strategies work better on the summaries of types MIXED and FREQUENT and are less effective on summaries of type RARE since the sampling strategies prefer more important concepts over less important ones.

Our system also works reasonably well on the original summaries from the DUC06 and DUC07 corpora, but the scores are much lower (see right-most plot of Figure 5). Yet this is not caused by the algorithm or sampling strategies but by the fact that the gold summaries of the DUC corpora are abstractive and not extractive (i.e., summaries are not composed of full sentences of the input documents but hand-written by users). This is in contrast to the summaries used for the left three plots of Figure 5 which are extractive, leading to a higher upper bound for the ROUGE2 metric.

Finally, a clear difference between the TOP-K and STRATIFIED sampling strategy can be seen in the results for Exp. 2b (Focused vs. Wide) as shown in Figure 6. Here, the STRATIFIED strategy shows a clear benefit over TOP-K when used on a WIDE summary (left plot) instead of FOCUSED ones (right plot) at least in the first few iterations. The intuition is that STRATIFIED sampling is better able to include sentences required for the breadth of WIDE summaries. The reason why TOP-K performs better in the latter iterations is that the sampling is more efficient (and covers also the desired breadth) when enough user feedback was collected. In future, we thus want to look into combinations of TOP-K and STRATIFIED sampling.

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.

REFERENCES

[1] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*, pages 670–680. ACL, 2017.
[2] 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.
[3] 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.
[4] 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.
[5] Avinesh P. V. S., B. Hättasch, O. Özyurt, C. Binnig, and C. M. Meyer. Sherlock: A system for interactive summarization of large text collections. *Proc. VLDB Endow.*, 11(12):1902–1905, 2018.
[6] 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.
[7] A. Nenkova and K. McKeown. Automatic summarization. *Foundations and Trends® in Information Retrieval*, 5(2–3):103–233, 2011.
[8] Y. Park, J. Qing, X. Shen, and B. Mozafari. BlinkML: Approximate Machine Learning with Probabilistic Guarantees. In *SIGMOD*, 2019. (to appear).
[9] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543. ACL, 2014.