Guiding Users by Dynamically Generating Questions in a Chatbot System

Jannis Pilgrim[1], Jakob Kemmler[1], Moritz Wassmer[1], Silvio Echsle[1] and Andreas Lommatzsch[2]

[1]Technische Universität Berlin, Straße des 17. Juni 135, D-10623 Berlin, Germany
[2]DAI-Labor, TU Berlin, Ernst-Reuter-Platz 7, D-10587 Berlin, Germany

Abstract

Chatbots efficiently support users in finding relevant answers in complex domains. They aggregate data from different sources and provide information in an interactive dialog. In a conversation, chatbots mime human experts providing information in well-consumable pieces. They try to guide users towards predicted information needs. One challenge for chatbots consists in generating questions if user inputs are ambiguous or incomplete. Computing good counter-questions requires an understanding of the user's intentions and a good structuring of the data to provide the needed information in a suitable format.

In this work we present a solution for generating clarification-questions based on dynamic data collections applying semantic clustering and flexible questions trees. We optimize and evaluate our approach for a chatbot tailored for answering questions related to services offered by the local Public Administration. We show that our approach efficiently helps users to find the relevant information in a natural conversation avoiding long lists for potentially interesting search results. The approach is based on a data enrichment and knowledge extraction pipeline that enables the adaptation of the components to different knowledge sources and the specific requirements of new domains.

1. Introduction

The rapid advances of NLP techniques in the last years and the big popularity of social media chat systems have led to a growing interest in chatbots. Chatbots mime the behavior of a human expert and provide relevant information and answers to users in suitable "pieces". Compared with complex web documents or long lists provided by search engines, chatbots guide users in a natural dialog to the needed information.

Providing an adequate answer to a complex user question is a challenging task, since user questions can be related to a huge range of topics and aspects. Moreover, user questions are often imprecise and ambiguous due to limited knowledge of the domain. Thus, a chatbot must determine the most relevant information based on the context and the knowledge about the user (making sure that all potentially relevant cases are considered). The computation of potentially relevant responses can be efficiently done based on Information Retrieval methods, optimized

LWDA'22: Lernen, Wissen, Daten, Analysen. October 05–07, 2022, Hildesheim, Germany
jannis.pilgrim@campus.tu-berlin.de (J. Pilgrim); jakob.kemmler@campus.tu-berlin.de (J. Kemmler); moritz.wassmer@campus.tu-berlin.de (M. Wassmer); silvio.echsle@campus.tu-berlin.de (S. Echsle); andreas.lommatzsch@dai-labor.de (A. Lommatzsch)

of finding potentially relevant information in large data collections. Determining the response best-fitting the user's intention usually requires additional information, which chatbots must collect by asking the right questions. The generation of good questions requires a precise understanding of the user question, a deeper analysis of potentially matching answers, and NLP techniques for generating questions and understanding given text snippets.

In this paper we analyze the scenario of optimizing a chatbot tailored for providing answers related to the public services of a major German city. In our scenario, citizens need information about the services offered by the public administration (e.g. how to get a residential parking permit or how to get a passport for babies). The conversation is usually started by the user with an initial question. Based on the question, the chatbot predicts potentially relevant services. In order to find the demanded information, the chatbot generates questions optimized for reducing the ambiguity in the user question and to reach the requested information with a minimal number of steps. We develop a component that deploys language models and dynamic decision trees for guiding the user to get the intentionally demanded answers.

The remainder of this paper is organized as follows. In Section 2 related research is summarized along an explanation to how they contributed to our work. Section 3 describes our approach and the structure of the underlying data. Subsequently, the evaluation of our approach is presented in Section 4, through quantitative and qualitative analysis of the chosen methods and developed chatbots. Finally, Section 5 discusses the accomplishments of this paper and provides an outlook on future work.

2. Related Work

In this section we review prior work on asking clarifying questions for conversational Information Retrieval (IR) and related methods. We first look at the usefulness of clarifying questions; then we review related approaches for systems asking clarifying questions. We conclude this section by analyzing clustering methods for IR and reviewing literature on decision trees for potentially relevant knowledge extraction methods that can be used in generating clarifying questions.

Usefulness of Clarification Questions Presenting information on small screen devices or in voice-only situations is challenging. Information access can be improved by systems that actively support interaction with the user [1]. It has been shown, that users like to be asked for clarification [2, 3]. Information obtained by clarifying questions can make huge improvements in terms of retrieval performance [4].

Clarification Question Models Zamani et al. [3] proposed three models to generate clarification questions and candidate answers for open-domain web search. The authors combined a rule-based slot filling model using question templates, a supervised model, and a reinforcement learning-based model. By aggregating huge collections of log data, weak supervision signals from query reformulation data are extracted and used for improving the models. The approach is not applicable in our scenario due to the small amount of log data that could be used for query reformulations learning.

Rosset et al. [5] built 2 conversational question suggestion models based on a BERT-based ranker and a GPT-2-based generator. They trained the ranking model in a multi-task fashion, mainly on weak supervision labels obtained from past users behavior like clicks on "People Also Ask" (PAA) Panes but also on human annotated relevance labels. Their natural language generation model is trained on the PAA questions which were clicked after the user issued a query. In our scenario a PAA approach can not be used due to the lack of sufficient logged existing user questions.

Aliannejadi et al. [4] collected a dataset named "Qulac" by crowd-sourcing to foster research in Clarifying Questions in IR for open-domain. They proposed a conversational search system that selects a clarifying question from a pool of questions and ranks stored documents based on the users answer. The researchers split up the task in question retrieval, selection and document retrieval. This approach requires predefined clarifying questions as well as query-question-answer-target mappings to train all components of the system. Due to the resource constraints of this project, an equivalent dataset cannot be generated, making this approach a bad fit for our use-case.

Datasets Even though a lot of datasets related to conversational search exist [6, 4, 7, 8, 9, 2, 10] most of them are either too domain-specific or not suitable regarding their structure to make use of them for our task. In addition, most of the datasets are in English language while our domain lies in German public administration language, which involves a lot of unique words. Therefore, we found those datasets not to be helpful for solving our problem like for example transfer learning existing models for our use case.

Clustering Text For an efficient refinement of a set of potential topics, the resources must be categorized first. This categorization might be based on the annotations provided by the dataset, which mostly follows a textual format. Following, we will discuss two papers, investigating different aspects of the concept of document clustering.

Leouski and Croft [11] compared different clustering techniques for analyzing textual retrieval results. They found good success using agglomerative hierarchical clustering algorithms in combination with frequency-based embeddings. Additionally, their results showed that human oriented evaluation should be preferred over an artificial one.

Mohammed et al. [12] analyzed document clustering by comparing two strategies based on a variety of popular evaluation methods. The researchers present an approach based on semantic embeddings in combination with a density-based clustering algorithm (DBSCAN) to outperform a frequency based embedding in combination with K-Means, especially on large datasets.

Discussion Most of the related works have in common that data and feedback signals were either crowd-sourced, or obtained by creating weakly supervised labels with the help of massive log data. Therefore they were able to train models with supervised methods. In many conversational IR settings outside of major web search industry, there is most likely not sufficient data for weak supervision available, or there are not enough resources to obtain data (as in our use-case). Approaches built upon clustering methods or decision tree-based keyword selection combined

with question templates may tackle these problems, since they do only require keywords for the documents to be retrieved. Therefore, it is worthy to investigate the proposed approaches.

3. Approach

In this section we give a comprehensive explanation of the three approaches developed in the context of this paper. First, we explain the dataset at the basis of our approaches. Then, a description of the fundamental algorithm, shared by all approaches is provided. After that, we give a detailed description of the implemented approaches as well as the technologies used.

3.1. Data

The basis of our approaches is an annotated dataset provided by the city of Berlin[1]. The dataset consists of 881 descriptions of services offered by the city administration, such as the renewal of a personal ID card or getting a residential parking permit. Each service entry is created by a human expert ("editor") with a list of keywords describing the respective service. These keywords consist of nouns, verbs, and numbers. A simplified example is shown in Table 1.

Table 1
The table shows an example which keywords are assigned to offered services

Service	Keywords
Parking ID Application	Apply, Parking
ID card Application	Apply
Parking ID Lost	Lost, Parking
Info about Pet ID Card	Pet
Change Address on ID Card	Address

3.2. General Approach

Our three approaches share the first step with the chatbot in place - the first user interaction in the form of an initial question. The user question is sent to an Apache Solr[2] server, providing an efficient full-text access to the aforementioned dataset. Given the query, the server retrieves a list of relevant resources. This list is iteratively refined until the desired resource is found. This refinement process consists of four steps.

First, all relevant documents are categorized and grouped based on their respective keyword annotations. Then, based on a heuristic, one group is selected and a superior term, representing all resources in that group, is inferred. Based on this superior term, a binary counter question using a question template is constructed and presented to the user (*Does your question revolve around topic X ?*). When the user answers affirmative, all resources but the ones contained in the selected group are removed from the list of possible resources. Otherwise, all resources

[1] https://service.berlin.de/
[2] https://solr.apache.org/

contained in the selected group are removed. This procedure is repeated until only one resource is left which is then presented to the user as the final answer.

The implementation of this algorithm raises two main challenges: the categorization of the resources and the inferences of the superior term describing the selected group. Following, we give a description of three approaches tackling these challenges using different strategies and technologies.

3.3. Service Clustering

The first approach, we name *Service Clustering*, is based on the applied method of grouping resources. For a visual depiction, see Fig. 1. After the initial list of relevant resources is retrieved from the indexed dataset, resources are categorized through clustering. To enable this, a meaningful representation is needed to make resources directly comparable to each other. This meaningful representation is created by encoding the keyword annotations for each resource using a text embedding. The embedding that was used is TF-IDF-based, as annotations commonly overlap between resources. The TF-IDF embedding emphasizes keywords that are distinct between resources. As those keywords carry the most information for differentiating between resources, this embedding provides vectors optimized for our setting. For clustering on the resulting vector representations, two algorithms were tested: K-Means and DBScan.

After clustering, one cluster is selected for generating a counter question. This selection should be made such that the information gained from the user is maximized, independent of the answer. Here, always picking the largest cluster is best strategy as the number of resources that can be eliminated is maximized.

Having selected the largest cluster, a superior term needs to be inferred. To achieve this, a semantic embedding was used. Semantic embeddings encode words such that the distance between their respective vectors corresponds to the semantic similarity between the words. Such a representation was created for every keyword contained in the annotation of at least one resource in the selected cluster. For this, the large German model[3] from the Spacy library was used. As the annotation vocabulary is highly domain specific, a reliable semantic embedding based on the used language models could not be computed for all potential relevant keywords. For the inference only keywords were used for which an embedding could be created. After the encoding step, the superior term was selected as the keyword whose representation has the smallest summed up distances to all other representations. This keyword forms the cluster centroid and is semantically closest to all words in the cluster, thus representing it best. As the measure of distances between vectors, cosine similarity was used.

Clustering Algorithms

For clustering the services, two algorithms were tested: K-Means [13] and DBSCAN [14]. The following gives a short description on the thought process behind this decision. Those two algorithms are commonly used in different problem settings surrounding text clustering [15, 16, 17]. As both are based on different ideas and therefore come with varying drawbacks, they are often compared to each other to find the best performing approach in a certain domain [18, 19].

[3] https://spacy.io/models/de#de_core_news_lg

K-Means K-Means is one the most popular clustering algorithms with a variety of applications, including document classification. In K-Means, the number of desired clusters needs to be specified. With the binary question template used in this approach, this is beneficial. In each iteration the maximum percentage that the set of relevant documents can be guaranteed to be pruned, is 50%. This is the case when the clustering results in two equally sized clusters. No matter the user's answer, half of the resources can be eliminated, resulting in a logarithmic conversion speed. K-Means allowing to specify the number of clusters to two might give a good approximation of these ideal conditions.

On the other hand, large clusters come with a significant drawback. As the number of resources per cluster increases, the complexity of finding a representative superior term increases. This might result in imprecise counter questions and therefore an error prone retrieval performance.

DBScan DBScan is a density-based algorithm, commonly used in the context of document classification. In contrast to K-Means clustering, DBScan does not require a fixed number of clusters. Instead a parameter *epsilon* is defined, specifying the maximum distance between two data points for them to be considered to be in one cluster. The methods results in a dynamically adapted number of clusters. That allows us to control the in-cluster similarity and to facilitate the inference of superior terms. In our experiments we found an epsilon of 1.3 to result in the best quality of superior terms. A high in-cluster similarity also results in smaller clusters and therefore a larger number clusters. This is likely to limit the number of resources that can be eliminated in each iteration, resulting in a slower conversion speed.

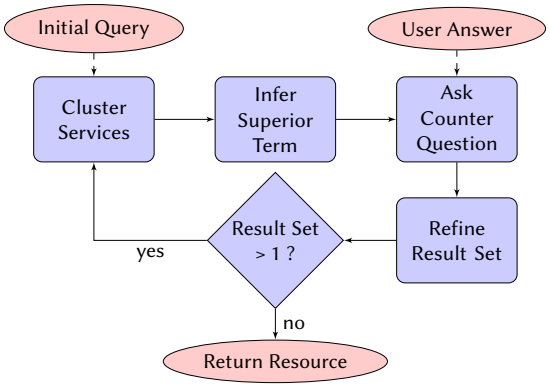

Figure 1: Visual depiction of the Service Clustering approach.

3.4. Question Tree

The next approach, we call the *Question Tree* approach, is based on the common decision tree algorithm ID3 [20]. Contrary to its usual usage in classification it is herein applied to the topic of information retrieval. It is constructed on run-time for every single query. Each node represents a question to the user and the target variable is the service name, implying that purity is reached only if the sample size in that branch is one.

Table 2
Keywords transformed to decision variable

Service	Address	Apply	Lost	Parking	Pet
Parking ID Application	0	1	0	1	0
ID Card Application	0	1	0	0	0
Parking ID Lost	0	0	1	1	0
Info about Pet ID Card	0	0	0	0	1
Change Address on ID card	1	0	0	0	0

The keywords annotation (Table 1) is used as a decision variable for the tree. Each keyword is mapped as a Boolean variable, indicating that a service either has it in their keyword list (equals 1) or not (equals 0). This results in a $n \times m$ matrix, where n is the number of services in a result set and m the number of keywords associated with any of these services. The result of the transformation is shown in Table 2.

Fig. 2 depicts a fully constructed tree for the fictitious example of an initial query of "identification" originating from the just-seen tables. Each node, in orange, represents a question about the keyword in its title. The heuristic of finding the group (the keyword) to ask the user about is choosing the variable that maximizes information, which is a greedy approach. The representation is always the keyword itself.

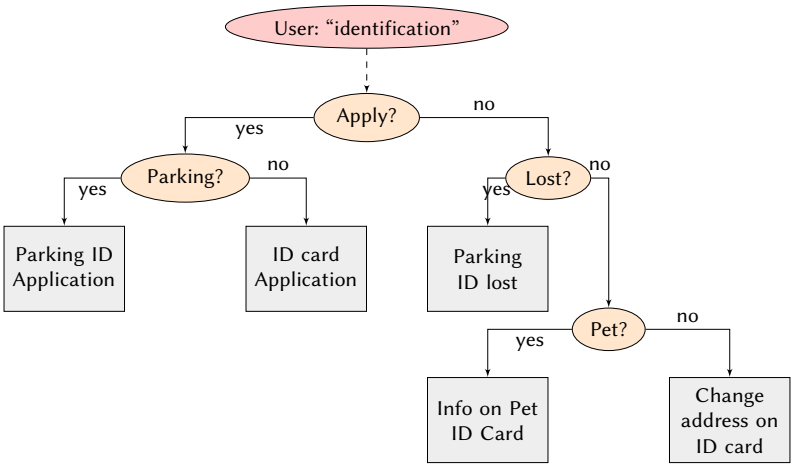

Figure 2: Graph visualizing all different possible paths after the fictional initial query of "identification". Nodes (orange) represent question "Does it have something to do with X?", grey boxes describe the assigned services (answers).

Due to the keywords being annotated by human annotators, imperfections could be found. These imperfections came in the form of different keywords that carry the same semantic information. This leads to unwanted side effects as connections between corresponding resources could not be made. In some cases, this makes for a frustrating user experience as sequential counter questions might ask for information already provided by the user. Additionally, this artificially blows up the decision tree. These mistakes in the annotation are of four categories: synonyms, different spelling of the same word, different grammatical surface forms of the same

word, words very close in semantic information.

To tackle this problem, we grouped these "similar" words to find a representation for each group. In order to find these groups, an initial clustering was applied, before the other steps were executed. As the number of semantically unique words and therefore the number of clusters was not known beforehand, DBSCAN [21] was used again. We set epsilon to 0.2 to ensure a high semantic similarity in each cluster. SpaCy's[4] large German model is used for defining a semantic embedding; the cosine similarity in the vector space is used for computing the similarity. Fig. 3 shows the program flow of the chatbot using the question tree approach.

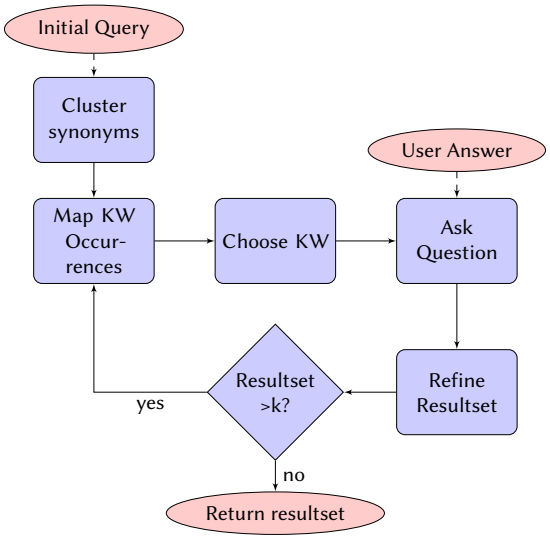

Figure 3: Graph visualizing the chatbot program flow using the question tree approach

4. Evaluation

In this section we describe our evaluation procedure and present the results obtained from the different approaches.

4.1. Quantitative Analysis

Procedure The main objective of the chatbot is to help users in finding the desired information quickly and to suggest only the service fitting the users intent. To ensure reproducibility and comparability in the evaluation, user interactions were simulated while logging of various measures is performed in the background. The test dataset consists of about 6,500 real user dialogues. We connected every initial query q to a service the user has clicked on s at some point of their dialog with the chatbot and assumed this to be their actual search intent, e.g. ground truth. As can be seen in Algorithm 1, the result set R initially returned from the Solr

[4] https://spacy.io/

system is iteratively refined based on simulated answers to the chatbots question. We reduced complexity by the assumption that users are always able to answer all the questions correctly.

The evaluation is biased towards actual chatbot usage, as only 363 of the total 881 services could be mapped to the initial user queries.

Algorithm 1 Evaluation Procedure Pseudocode

```
for (q, s) in Q do
    R ← solr.getResults(q)
    while length(R) ≥ k do
        question ← chatbot.getQuestion(R)
        answer ← findCorrectAnswer(R, question, s)        ▷ simulate correct user answer
        R ← chatbot.refineResultset(R, question, answer)
    end while
end for
```

If the initial query yields a result set of any length, all three approaches are guaranteed to find the intended user services, respectively a result set of length k. This is due to the fact that the Service Clustering approaches re-cluster at every iteration and that every combination of keywords is unique. We introduce two different measures to compare the approaches:

- **Mean Turns:** Mean turns needed to find a service
- **Mean Information Gain:** Mean information gain of the answer to a question

The Information Gain (IG) of an iteration (Choose Question-Answer-Refine) n is defined as the difference between the natural logarithm of the length of the result set R before and that after the answer, depicted in following equation:

$$IG(n) = \ln(length(R_{n-1})) - \ln(length(R_n)) \tag{1}$$

Results We evaluate the different approaches quantitatively with the just described procedure and iterate until $k = 1$. Fig. 4 shows the distribution over how many turns are needed until a conversation converges and Table 3 holds all results of the quantitative analysis. The distributions are varying significantly. SC KMeans takes with over five turns on average the longest to converge but is also the most prone to outliers (e.g. services that might be hard to find) and very evenly distributed. SC DBSCAN is converging faster on average but includes some conversations taking over 25 turns. The Question Tree approach using DBSCAN outperforms all other approaches in terms of speed with less than four turns and an information gain of 0.950 per question on average. None of the three approaches dominate all measures - however the Question Tree seems to more suited for the use case, at least according to the quantitative approach. In the following, we will highlight the qualitative point of view.

4.2. Quality of Questions

In this section we analyze the quality of the generated questions focusing on the keywords chosen by the approaches.

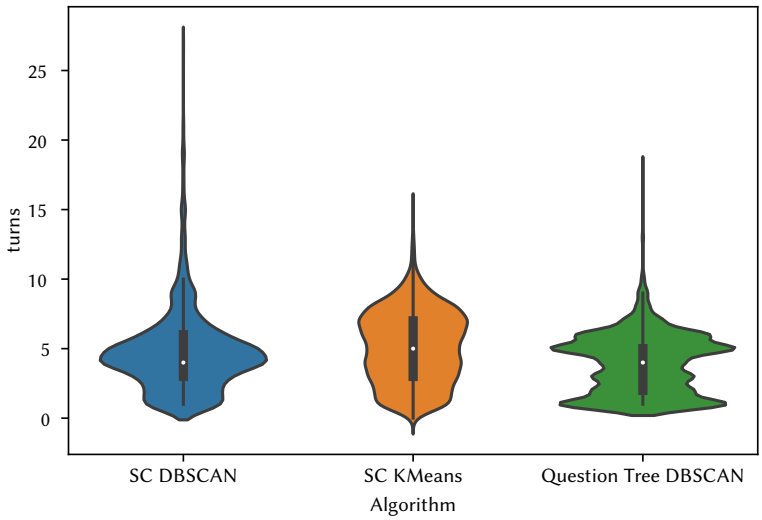

Figure 4: Distribution over turns needed till service found. The figure shows that SC DBScan has the highest variance. On Average, the Question Tree DBScan needs the smallest number of turns.

Table 3
The table shows the obtained performance scores for the implemented methods.

Algorithm	Mean Turns	Max Turns	Information Gain
Service Clustering DBSCAN	4.69	27	0.904
Service Clustering KMeans	5.07	15	0.826
Question Tree DBSCAN	3.88	18	0.950

The keywords used as the basis for the generated questions should come from the user's vocabulary (to ensure that the user knows the terms) and the keywords should be unique and simple (to minimize the ambiguity). When analyzing the approaches, an interplay between two characteristics of chatbots can be observed. Chatbots, whose largest cluster comprises almost 50% of the services converge particularly quickly on a result set. Clustering methods that are able to cluster around 50% of the services in each iteration also halve the result set by about 50%, regardless of the user's response. Here one can observe parallels to bisection, where the interval width is halved with each step, to perform in a runtime of $O(log(n))$. The second relevant property is the representativeness of the keyword. Some approaches tend to find very general keywords or ask for the same keyword multiple times. This can be attributed to the fact that too large clusters have been formed, for which it is difficult to find a common keyword. The qualitative experiments confirm our assumption that chatbots with higher convergence times ask qualitatively better questions. The specific characteristics of the algorithms have been studied at several examples. The main observations are explained in the subsequent paragraphs.

Question Tree The question tree is one of the faster converging approaches. This can be explained by the fact that the focus of the algorithm is on selecting particularly good keywords and the clusters result from the user's decision.

Service Clustering with k-means The approach of service clustering with k-means as the clustering method was convincing with particularly short dialogues. The reason that this approach converges so quickly to a solution can also be attributed to the cluster size of the largest cluster being close to 50% of the result set. This forces in some situations that services are assigned to large clusters, but have no influence on the keyword in these clusters. These decisions ultimately make for less precise queries due to an inaccurately chosen keyword.

Service Clustering with DBSCAN The third approach uses the clustering algorithm DBSCAN. This algorithm determines the number of clusters at runtime, which allows it to create new clusters according to the number of topics. By adjusting the epsilon parameter accordingly, this effect can be controlled. This control allows to keep an eye on the convergence speed as well as on the quality of the questions.

Overall, our analysis shows that the Question Tree approach provides the best questions.

5. Conclusion

In this work, we presented a solution for generating counter questions based on dynamic data collections. We developed and evaluated three approaches combining semantic clustering and decision trees. The methods have been optimized to the specific requirements of the chatbot for the German public administration. Our experiments show that our method generates reasonable questions, effectively guiding users to desired resources in an intuitive conversational style. Our solutions provide the basis for a well-working interactive information retrieval system. Our approach can also be applied to similar scenarios, since it can be used with text collections of answers or documents. With our findings, we contributed to the research in chatbot systems and information retrieval. As future work we plan to improve the generation of questions to improve the natural soundness and to better adapt to the context specific language style.

Acknowledgment

We thank the ITDZ Berlin for supporting the development of the chatbot framework.

References

[1] W. B. Croft, The importance of interaction for information retrieval, in: Procs. of the 42nd Intl. ACM SIGIR Conf., ACM, NY, USA, 2019, p. 1–2. doi:10.1145/3331184.3331185.

[2] J. Kiesel, A. Bahrami, B. Stein, A. Anand, M. Hagen, Toward voice query clarification, 2018. URL: https://dl.acm.org/doi/pdf/10.1145/3209978.3210160.

[3] H. Zamani, S. Dumais, N. Craswell, P. Bennett, G. Lueck, Generating Clarifying Questions for Information Retrieval, ACM, NY, NY, USA, 2020, p. 418–428.

[4] M. Aliannejadi, H. Zamani, F. Crestani, W. Croft, Asking clarifying questions in open-domain information-seeking conversations, 2019.
[5] C. Rosset, C. Xiong, X. Song, D. Campos, N. Craswell, S. Tiwary, P. Bennett, Leading conversational search by suggesting useful questions, in: The Web Conference '20, 2020.
[6] H. Zamani, G. Lueck, E. Chen, R. Quispe, F. Luu, N. Craswell, Mimics: A large-scale data collection for search clarification, in: Proc. of the 29th ACM CIKM, CIKM '20, ACM, New York, NY, USA, 2020, p. 3189–3196. doi:10.1145/3340531.3412772.
[7] C. Qu, L. Yang, W. B. Croft, J. R. Trippas, Y. Zhang, M. Qiu, Analyzing and characterizing user intent in information-seeking conversations, in: The 41st Intl. ACM SIGIR Conf., ACM, 2018. doi:10.1145/3209978.3210124.
[8] R. Lowe, N. Pow, I. Serban, J. Pineau, The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems (2015). doi:10.18653/v1/W15-4640.
[9] F. Radlinski, K. Balog, B. Byrne, K. Krishnamoorthi, Coached conversational preference elicitation: A case study in understanding movie preferences, in: Procs. of the 20th SIGdial Meeting on Discourse and Dialogue, ACL, Stockholm, Sweden, 2019, pp. 353–360.
[10] E. Choi, H. He, M. Iyyer, M. Yatskar, W.-t. Yih, Y. Choi, P. Liang, L. Zettlemoyer, Quac : Question answering in context, 2018. URL: https://arxiv.org/abs/1808.07036. doi:10.48550/ARXIV.1808.07036.
[11] A. V. Leouski, W. B. Croft, An evaluation of techniques for clustering search results, Technical Report, 1996.
[12] S. M. Mohammed, K. Jacksi, S. R. M. Zeebaree, Glove Word Embedding and DBSCAN algorithms for Semantic Document Clustering, in: Intl.Conf. on Advanced Science and Engineering, 2020, pp. 1–6. doi:10.1109/ICOASE51841.2020.9436540.
[13] E. W. Forgy, Cluster analysis of multivariate data: efficiency versus interpretability of classifications, biometrics 21 (1965) 768–769.
[14] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise., in: kdd, volume 96, 1996, pp. 226–231.
[15] C. Xiong, Z. Hua, K. Lv, X. Li, An improved k-means text clustering algorithm by optimizing initial cluster centers, in: 7th Intl. Conf. on Cloud Comp. and Big Data, 2016, pp. 265–268.
[16] R. G. Cretulescu, D. Morariu, M. Breazu, D. Volovici, Dbscan algorithm for document clustering, Intl. Journal of Adv. Statistics and IT&C for Economics and Life Sciences 9 (2019).
[17] R. N. G. Indah, R. Novita, O. B. Kharisma, R. Vebrianto, S. Sanjaya, T. Andriani, W. P. Sari, Y. Novita, R. Rahim, et al., Dbscan algorithm: twitter text clustering of trend topic pilkada pekanbaru, in: Journal of physics, volume 1363, IOP Publishing, 2019, p. 012001.
[18] M. A. Ahmed, H. Baharin, P. N. Nohuddin, Analysis of k-means, dbscan and optics cluster algorithms on al-quran verses, Intl. Journ. of Adv. Computer. Science and Apps. 11 (2020).
[19] D. Xu, Y. Tian, A comprehensive survey of clustering algorithms, Annals of Data Science 2 (2015) 165–193.
[20] Data mining, practical machine learning tools and techniques, in: I. H. Witten, E. Frank, M. A. Hall, C. J. Pal (Eds.), Data Mining, 4th ed., Morgan Kaufmann, 2017, pp. i–iii. doi:https://doi.org/10.1016/B978-0-12-804291-5.00014-3.
[21] S. Mohammed, K. Jacksi, S. Zeebaree, A state-of-the-art survey on semantic similarity for document clustering using glove and density-based algorithms, Journal of Electrical

Engineering and Computer Science 22 (2021) 552–562. doi:10.11591/ijeecs.v22.i1.