Dynamic Faceted Search for Technical Support Exploiting Induced Knowledge

Nandana Mihindukulasooriya(✉), Ruchi Mahindru, Md Faisal Mahbub Chowdhury, Yu Deng, Nicolas Rodolfo Fauceglia, Gaetano Rossiello, Sarthak Dash, Alfio Gliozzo, and Shu Tao

IBM Research, T.J. Watson Research Center, Yorktown Heights, NY, USA
{nandana.m,nicolas.fauceglia,Gaetano.Rossiello}@ibm.com
{rmahindr,mchowdh,dengy,sdash,gliozzo,shutao}@us.ibm.com

Abstract. IT support is a vital and integral part of technology adoption. Conventionally, IT support service providers heavily rely on human effort and expertise to respond to user queries. Given the cost-benefit and 24×7 availability for answering user questions, Virtual Assistants (VA) are highly applicable in the technical support domain. In this paper, we describe a novel methodology for building interactive virtual assistants for IT support using Dynamic Faceted Search (DFS). Given a question, dynamic facets are generated automatically, enabling the user to refine and narrow down their intent. To do so we leverage knowledge automatically induced from textual content and existing Semantic Web resources such as Wikidata. Such knowledge is then used to dynamically generate facets interactively based on the user's responses as shown in the demo video (https://ibm.box.com/v/iswc2020-dfs). The experiments on two real-world datasets in the IT support domain show the effectiveness of DFS in refining the user's queries and efficiently identifying possible solutions to their technical problems.

Keywords: Faceted search · Knowledge induction · IT support · Virtual assistant

1 Introduction

IT support is committed to help customers identify solutions for issues occurring in hardware and software products, and suggest respective solutions. In this domain, a large amount of support documentation is available, typically consisting of user manuals and troubleshooting (*how-to* or *what-is*) documents. Such documentation contains solutions to common problems that the customers may face with the products.

Information retrieval methods, *e.g.* keyword-based search, could be applied in seeking the relevant document which provides the solution to user's query. However, this strategy is not adequate in domains like IT support, where the

N. Mihindukulasooriya and R. Mahindru—Equal contributions.

J. Z. Pan et al. (Eds.): ISWC 2020, LNCS 12507, pp. 683–699, 2020.
https://doi.org/10.1007/978-3-030-62466-8_42

complexity of problems often leads to several challenges. First, user's problems are often too complex to be expressed with a single query, as the user may be observing several different issues at the same time, e.g. `fan making noise` and `battery not charging` can occur simultaneously. In some cases, problems observed could be related but in other cases they may be completely unrelated. Second, user may craft the question based on their observations, while the content in technical documentation is formally written by Subject Matter Experts (SMEs). Therefore, leaving a tremendous gap in wording and rhetorical structure used for asking a question versus content writing. Third, a user may not know a priori the context (relevant elements) that should be provided for the system to efficiently find the target result.

Faceted Search (FS) [15] is a prevalent technique for interactive information retrieval, e.g. in e-commerce. It involves augmenting a document retrieval system with facets to narrow down search results[1]. Users looking for IT support often find it challenging to formulate complete query for search. FS can `guide` users to refine initial queries and `navigate` to the target result. Traditional facet generation approaches present several drawbacks. Documents must be tagged with an existing taxonomy, adding overhead in content curation and management. Moreover, static facets are not based on the matching documents or queries. DFS overcomes such limitations [7].

In attempting to solve the limitations discussed above, we propose VA for IT Support based on DFS. Using DFS, our system allows refinement of the initial user query, in an interactive way, by presenting to the user multiple choices (for augmenting and refining the query) in the form of facets. Use of VAs to help answering customer issues is on a rise [6], as it is becoming a necessity for service providers to provide 24×7 IT support at a lower cost.

Typically, building a VA for a specific domain requires a considerable effort as it often relies on curated knowledge consisting of hand-crafted databases, problem determination flows and predefined question/answer pairs developed by SMEs. Such effort could typically range from anywhere between a few weeks to months which hinders automatic and rapid initialization of VAs to new domains.

Additionally, our experience shows that typically support issues cover 10–20% common questions and, remaining are long-tail questions addressed using documents. VA uses dialog technology for the common questions, redirecting the long tail to regular search on technical documentation. In this work, we address the long tail questions using Dynamic Faceted Search. DFS provides an interactive search experience to guide the user to form a more complete query.

In the absence of any kind of annotated training data (which is common for many IT service providers), exploiting the textual content of technical documentation in an unsupervised (or minimally supervised) manner is the most viable choice to avoid the laborious effort from the SMEs and delay to market.

The main features of our proposed DFS based VA system are as follows:

– The facets are dynamically generated from the textual content (in IT support documents) during runtime. In other words, they are not static, *i.e.*,

[1] A VA capable of fully automatic dialog generation is out of scope for DFS in the context of this work.

pre-tagged for every document. Instead, they are dynamically generated based on the user's query and facets selected so far.
- Our system generates two kinds of facets, namely `flat facets` and `typed facets` (more details in Sect. 3.3). The system can be configured to show the kind of facets a user prefers.
- Unlike conventional faceted search systems (often used for e-commerce), our facets are not specific to particular items (e.g. "price" or "brand" products) or documents (e.g. "topics" or "categories" of a document). Such facets are not sufficient for all domains. Due to this flexibility, even though we are presenting our VA system here for the IT support use case, it can be easily adapted to other domains that share similar or subset of characteristics as IT Support domain.

In addition to the above, this paper has two additional contributions. Firstly, we propose an automatic evaluation setting to simulate human users. Secondly, we empirically show that our FS based approach improves the results of document retrieval of a popular IR based approach.

In the next section, we define the use case relevant in scope for information retrieval over textual documents. An overview of our system and the architectural design are described in Sect. 3. In Sect. 4, we present our experimental protocol with the discussion relative to the results. Section 5 presents an overview regarding the progress made by the community in conversational search. Finally, Sect. 6 concludes the paper along with directions in our research agenda.

2 Use Case

IT Support is contracted to deliver several business key performance indicators (KPIs), e.g. first-time-fix (FTF). FTF is a measure indicating, whether the provided solution resolved the user's issue in the first time, they contacted the support or used the troubleshooting tools (e.g. search, virtual assistant etc.) directly to resolve their problem. For any customer side support whether from technical domain or not, customer satisfaction is typically one of the most important business success metrics. Therefore, it is critical to increase FTF improve customer satisfaction. In fact, it was found that the average first response time for support tickets is 5.32 h[2].

In the IT domain, user questions typically fall in three major categories: (1) `troubleshooting`, e.g. *"no power"*, *"jextract utility OutOfMemory error"* or *"key is stuck"*; (2) `how-to`, e.g *"password reset"*, *"replace disk drive"*, *"secure DataPower MQ"*; (3) `what-is`, e.g. *"what version of java is supported in DataPower Appliance 6.0+"*, *"what feature does my Android have"*.

End users' problems, expressed with queries, are primarily biased by the terminology that they are familiar with. While the technical documents are written by experts and the technical description of the problem may be quite

[2] https://www.jitbit.com/news/255-lessons-learned-from-analyzing-7-million-customer-support-tickets/.

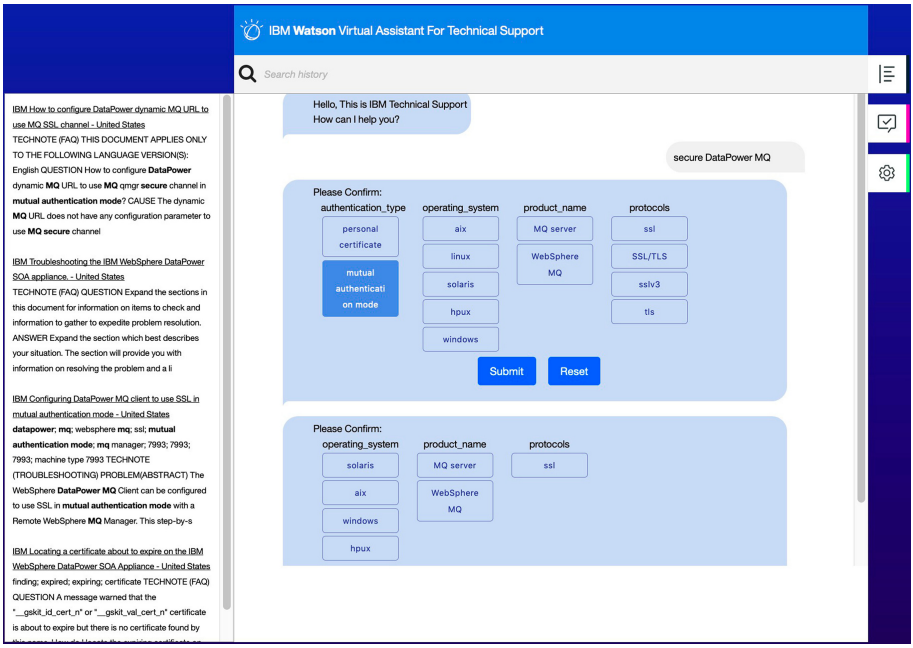


Fig. 1. Virtual assistant interface

remote from how the user may describe the issue. This disconnect in user's query and content along with complexity of technical problems often leads to several challenges, as discussed below.

- The user may express multiple problems in a single query. For example, *keyboard not typing* and *battery keeps discharging*. Here, there are two independent problems described. In another example, *ac adaptor faulty* and *battery is not charging*. In this case, *battery not charging* could be a side effect of *ac adaptor faulty*, hence there's a cause and effect relation.
- User may not know a-priori the context (relevant elements) that should be provided to efficiently find the target result. For instance, a user may experience that there is *"no audio"* output. There can be many different documents (*e.g.*, driver-related issues, connectivity issues, hardware issues) pertaining to a user query such as *"no audio"*. Such a query is under-expressed leaving a large room for the system to determine the precise answer document.
- User may craft the question based on their observations, while the content in technical documentation is formally written by Subject Matter Experts (SMEs). For example, user may express their query as *storage failure* while the document may express and describe the problem as *Hard Disk Drive Failure*, *Solid State Drive Failure*. Both *Hard Disk Drive* and *Solid State Drive* are storage related but neither document may directly use the word storage in them, while there could be many other unrelated documents using

the term *storage failure* in them but not necessarily related to resolving user's problem. Therefore, leaving a tremendous gap in wording used for asking a question versus content writing.

To address the challenges described above, our goal is to build a VA that given a user query, determines whether the user query is under-specified, contains multiple problems based on the rhetorical structure, or it is missing context.

In this paper, our focus is to tackle under-specified queries. Given the user query, VA fetches the relevant documents and presents them to the user, along with the facets. Thus interactively nudging the user to further refine their query, so that the augmented query can be used to retrieve more relevant documents (see Fig. 1). One can argue that this may also be accomplished to a certain extent by a non-interactive search system. Based on our observation from the real-world application, interactive VA helps reduce the cognitive burden of the user in coming up with a more complete query. And thus, guide them towards the target answer by presenting the contextually relevant facets. VA for technical support may cover other use cases[3], which are more close to dialog style (i.e. conversational) interaction, e.g. chatbots rather than search-oriented (long tail). In this paper, our focus is primarily on leveraging textual documents to 1) perform content driven and interactive user query augmentation, and 2) improve the ranking of the results retrieved.

3 End-to-End System Overview

We present an end-to-end system composed of two phases: 1) *Virtual Assistant Initialization (VAI)*, an offline process, which consists of two main steps – unsupervised knowledge induction and curation; and 2) *Virtual Assistant Runtime (VAR)*, also consists of two main steps, an online VA application for user interaction, integrated with the DFS component (see Fig. 2). In each phase, there are dedicated personas interacting with the system in the VAI and VAR phases.

3.1 Personas

The `two personas` that are involved in the end-to-end use case (described earlier) are – the `solution designer (SD)`, e.g. an SME, who is specialized in the problem troubleshooting and the required domain content, is interested in creating a VA and leads the VAI, and the `user` of the VA is involved in the VAR.

Solution Designer. The `SD` uploads relevant documents (which would be used later for resolving users' questions) in the system. The system automatically induces knowledge (e.g. domain specific terms, taxonomy, etc) from the documents. The SD can fine-tune the induced taxonomy using the smart spreadsheet

[3] For example, interactive problem diagnosis containing test and action steps; and process automation to invoke enterprise endpoints. for common questions.

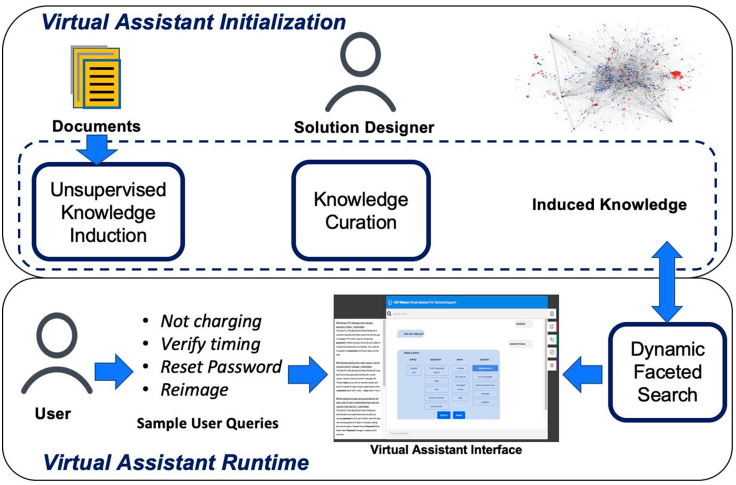


Fig. 2. End-to-end system overview

editor available in the system (see Sect. 4 in [10]). Lastly, the VA uses this curated knowledge together with other artifacts to generate facets dynamically to help the user refine their query.

Virtual Assistant User. The `user` persona uses the VA to find a solution to a problem that they are facing. In the IT domain, it could be either a support agent responsible for handling customer service requests, or it could be an end user directly interacting with the VA on the vendor's web-site.

Users of popular search engines, such as Google, generally have the tendency to formulate fragment queries [14] which are often under-specified. Therefore, by presenting dynamically generated facets relevant to the user's query, the VA can interactively solicit additional information from the user. Such information augments the query and enables the VA to retrieve updated results along with a new set of ranked facets. This interaction is repeated until the user is satisfied with the results presented and no further refinement is needed.

The VA interface drives the conversation and handles the user queries. Given a user query, it performs the tasks as shown in the *Facet Generation* component in Fig. 3. The refined facets and re-ranked search results are presented to the user in the VA after every interaction.

3.2 Virtual Assistant Initialization

This phase includes two main components as illustrated in Fig. 3:

- A component for automatic knowledge induction.
- A component for curating automatically induced knowledge where the induced knowledge is presented to the `SD` in a `smart spreadsheet` [10], so that they *quickly* remove spurious results to enhance it.

Unsupervised Knowledge Induction. The knowledge induction (KI) service trains term embedding models, extracts domain specific terms and induces taxonomies (types and their instances) from input documents. One of the embedding models trained is a type model (see Sect. 3.2 in [10]). The embedding models are used by all the components in Fig. 3. The state-of-the-art automatic taxonomy induction component uses an ensemble of a pattern-based approach, symbolic knowledge-based approach, and a neural network-based approach [8,10].

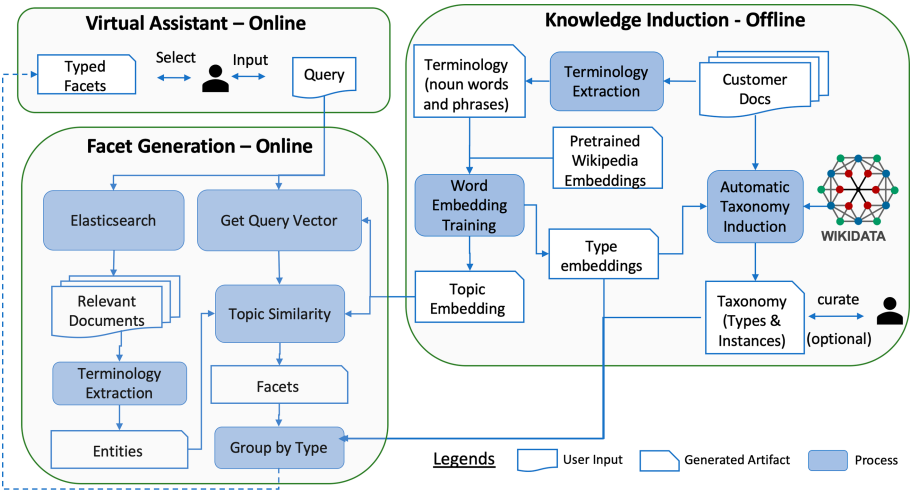


Fig. 3. Architectural overview

In the pattern-based approach, twenty-four *lexico-syntactic* patterns (e.g., "NP$_y$ is a NP$_x$"), aka Hearst-like patterns [11] are used for *is-a* pair extraction. Then, several post-processing steps are carried out to clean the extracted *is-a* pairs by removing cycles and proper nouns as types [10] and to expand them using super and nested terms [5].

Figure 4 illustrates the usage of symbolic knowledge from Wikidata in knowledge induction. Wikidata is used to create a dictionary of pre-known *is-a* using *instance of* (P31) and *subclass of* (P279) relations. For each corpus, terminology is extracted as shown in Fig. 3 and linked to the entities in Wikidata. This process extracts an *intermediate domain taxonomy* from the dictionary. Finally, to mitigate any entity linking error propagation, we perform a cleaning step using cosine similarity (a) each *type* and *instance* and (b) between all *instances* of a given type. This produces a domain taxonomy for the given corpus.

In the neural network-based approach, we use Strict Partial Order Networks (SPON) [8], a neural network architecture comprising of *non-negative* activations and *residual* connections designed to enforce strict partial order as a soft constraint. The union of the *is-a* pairs extracted from previous two approaches are used as the input for the neural model. SPON finds the *is-a*

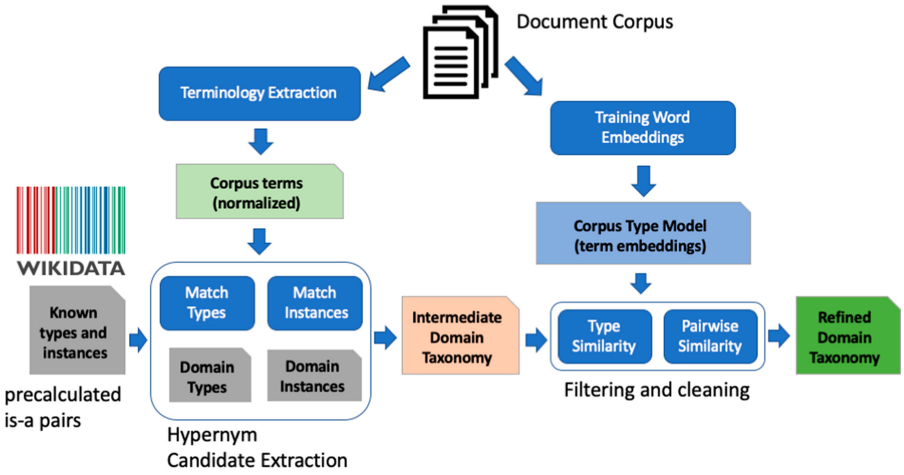


Fig. 4. Wikidata taxonomy generation

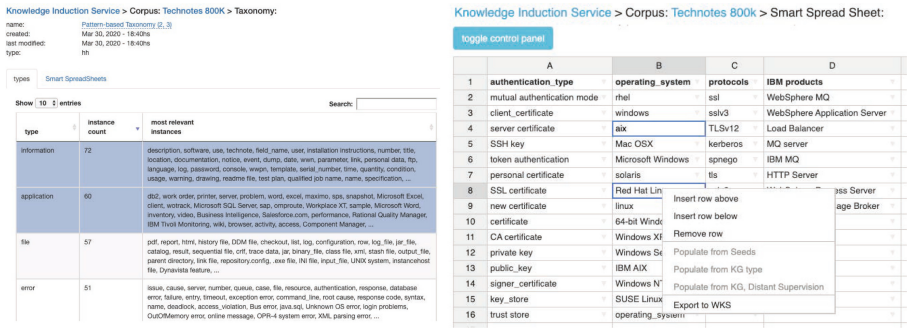
relationships among terms in the terminology that were not discovered by the previous two approaches. Please refer to [8] for the details.

This process of inducing a taxonomy automatically, given a corpus of natural language documents can reduce VA development effort from several weeks to a few hours, allowing scalable development of such VAs.

Knowledge Induction service allows solution designers to upload new documents and rerun the knowledge induction process to support evolving corpora. It will update the knowledge artefacts such as the terminology, embedding models, and taxonomies. Once updated, these new artefacts will be used seamlessly by `Dynamic Facet Generation` so that the knowledge from new documents will be used in DFS.

Knowledge Curation. As mentioned earlier, the `SD`, if needed, can curate the induced domain specific terms and taxonomy, to further refine them in a human-in-the-loop manner; based on the real-world use cases (with corpora up 800 K documents), this process can be generally performed in a few hours for a reasonably large corpora.

This curation process generally starts with presenting the `SD` with a list of types from the generated taxonomy. For making the inspection of `SD` efficient, the list can be sorted by the frequency they appear in the corpus or the number of the instances a type has. Once the relevant facets are selected, `SD` can export them to a `smart spreadsheet` (Fig. 5). In this spreadsheet, `SD` can fine-tune the instances in the types shown or add brand new types. Furthermore, `SD` can expand the instances using existing ones as seeds and finding the nearest neighbours of those in the type embedding model or by performing Wikidata lookups.



A.) Type Selection View B.) Smart Spread Sheet View

Fig. 5. Smart spreadsheet for knowledge curation

3.3 Virtual Assistant Runtime

This phase includes two main components as illustrated in Fig. 2 –

- A component with a chat interface (see Fig. 1). It receives user questions and shows the most relevant documents containing the answer.
- A component for facet generation and search (see Fig. 3). It dynamically generates facets based on information retrieval result for the query and induced knowledge.

Virtual Assistant Interface. First, the `user` uses the VA interface to submit the initial query. Next, the system submits the query to the dynamic facet generation component. As described in Sect. 3.3, these facets are generated from the terminology present in the corresponding search results of the initial query. We would like to emphasize that such content driven guidance is critical, as the user may not know off hand the terminology relevant to retrieve the target answer. The user selects one or more relevant facets to refine the query, hence helping the user find the most relevant document faster (Fig. 1).

For example, suppose the user has a problem with "*database password running on Microsoft SQL Server*". But they may under specify a query such as "*password issue*", which is typical for the support agents. In this case, the VA starts by returning a set of initial results based on the query, as shown on the left hand side of the interface. In addition, the VA nudges the user to refine their query, if needed. The user is able to further clarify the query as their problem is related to a *database error* running on *SQL Server*. Finally, the user finds the relevant document per the refinements.

Dynamic Facet Generation. We automatically generate facets from the document corpus and show the relevant facets dynamically per the semantic matching of the user query and search results (Fig. 3). As a first step, the user query

is submitted to the ElasticSearch[4] IR component, which returns a ranked list of documents as search results. As mentioned earlier, there are two types of facets described as follows.

Flat facets – This algorithm starts by extracting all domain specific terms, $\mathcal{E}$, present in the search results for the input query, which then are filtered by removing the ones – (i) present in the query, (ii) previously selected facets (if any), and (iii) whose topic similarity with respect to the query are below a threshold (default, 0.5). The remaining terms are then clustered based on the similarity score predicted by type embedding model. This is done to avoid near identical terms. The highest ranked term from each cluster is kept and they are ranked according to their topic similarity with respect to the query. The top n remaining terms are then returned as facets.

Typed facets – In this case, the system ranks each term in the induced taxonomy by topic similarity with respect to the query, looks for their corresponding types and collects weights for each type following a `k-nn` (k-nearest neighbours) based approach. This allows the system to select a certain number of types and rank them. Following that, the system looks for other terms (in the documents in search results for the query) that are similar to the terms in the type set selected in previous step. This is done by selecting the top `n` other terms if their type similarity is above a certain threshold. Finally, the top `k` types containing maximum top `x` facets per type are returned as typed facets.

4 Experiments

We evaluate our VA system through a quantitative and qualitative analysis. Quantitative evaluation aims to measure the impact of the proposed dynamic facet generation in retrieving the target (i.e. most relevant) document after the facet selection(s). We perform qualitative evaluation to investigate the quality and usefulness of those facets from the user experience perspective. A manual evaluation with SMEs is not viable for large amounts of data and for repeating experiments with different settings. Therefore, we have devised an automated evaluation for quantitative analysis using two datasets for experiments. Each dataset consists of document corpus and question answer pairs. The question answer pairs are actual questions posted by real-world users in forums and answer is link to the document in the corpus.

TechQA Dataset. The first dataset is the TechQA dataset [4] which contains real-world user questions posted on IBM DeveloperWorks[5] forums in the domain of technical customer support. This dataset is created by extracting forum questions with their accepted answers where a link to an *IBM Technote*, a technical

[4] https://elastic.co/enterprise-search.
[5] https://www.ibm.com/developerworks/community/forums/.

document, appears in the text. We use both the 160 question answer pairs in the development set and the full set of 610 question answer pairs for our benchmarks. The TechQA dataset also contains a corpus of 801,998 publicly available IBM Technotes documents.

Private Dataset. The second dataset is a private dataset, which contains a corpus of 4,000 technical troubleshooting documents from a Personal Computer domain. Additionally, it has 50 question answer pairs from forum corresponding to the same domain and product. This dataset comes from a real technical support customer engagement that we have used for internal evaluation purposes and due to confidentiality, the dataset cannot be shared publicly.

For each dataset, we use the documents to induce their own knowledge artifacts as described in Sect. 3.2. In our experiments, we only use the title of the forum post as the query to simulate under-specified user queries. Then we simulate the refinement of the query using the facets generated by the system in an interactive manner. The criteria used in the evaluation is how efficiently (with minimum number of clicks) a user can find the answer with the help of facets generated by our VA.

4.1 Quantitative Evaluation

In order to evaluate the impact of the facets, we compare our VA system against the vanilla ElasticSearch (i.e. baseline), which is used by our system as the IR component. We evaluate if the results are improved after the queries are augmented with the generated dynamic facets. We use four evaluation metrics: *Mean Reciprocal Rank (MRR)*, *Hits@1*, *Hits@5* and *Hits@10*. For *Hits@K* metrics, we share the ratio of the number of queries where the expected document is ranked within top-K results compared to the total number of queries.

User facet selection is simulated using an algorithm that we call ***O*racle**, discussed below.

1. First, the n-grams are extracted from the user query.
2. Second, the facet generation component is used to obtain the search results and generate facets with scores (either typed facets or flat facets).
3. Third, the Oracle first selects the top-K[6] facets (regardless the facets are typed or flat) based on the scores associated with relevance of each facet with the query. Here, by selecting top-K, we assume that in a live setting, a human user might not read more than the top-K facets.
4. Next, out of the top-K facets, the Oracle selects the best facet which retrieves the target document at the highest possible rank. Here, we make a simplified assumption that in a live setting a human user (*e.g.* SME) will be always able to identify the best facet (with respect to her query) among the top-K facets presented.

[6] K = 5 in our experiments.

Table 1 shows the performance comparison for the TechQA development dataset of 160 question answer pairs and full dataset of 610 question answer pairs. As shown in the table, both DFS approaches, i.e. search enhanced by flat facets and typed facets, have improved the baseline results for both cases by refining user queries. Interestingly, *DFS using typed facets* performs better than *DFS using flat facets*. We observe similar performance improvements in the private dataset as shown in Table 2. This underpins the value of our unsupervised induced taxonomy for organizing dynamically generated facets.

In case of VA use case, our target is to present the best possible document result in the top 5 (*Hits@5*) to improve user experience with the minimum number of facet clicks. Based on our interaction with the real-world users, we used 3 as the maximum number of rounds for facet selection (i.e. up to 3 user clicks to find the answer document). With the typed facets, we observe that the *Hits@1* and *Hits@5* are improved by up to 9% (as shown in Table 1). Additionally, as observed, flat facets follow the similar trend in improvement, where *Hits@1* and *Hits@5* are improved by 6% (as shown in Table 1). In the second data, similar hyper parameters were used and confirms the same observation in result improvements, as shown in Table 2).

4.2 Qualitative Evaluation

For the qualitative evaluation, we selected a sample set of random queries from the 160 queries. We manually inspected the facets using a *human-in-the-loop* approach. A facet is considered useful, if it (i) contains terminology (that is contextually related but not already mentioned in the user's actual query) from the fully specified query, and (ii) appears in the target answer document.

Table 3 shows a few fully specified queries, actual (under-specified) queries entered by the users in the VA, and the corresponding dynamically generated flat facets.

For example, in the first row of Table 3, the ideal query is *"How to configure SSL using mutual authentication in DataPower MQ client?"*, while the user may enter *"DataPower MQ client security"*. This under-specified actual query leaves room for fetching many generic documents containing "security" and "DataPower MQ".

Table 1. MRR and Hits@K results for the development set of 160 questions and the full set of 610 questions in the TechQA dataset using flat and typed facets.

	Dev 160 questions			Full 610 questions		
Metric	Baseline (ES)	DFS (Flat)	DFS (Typed)	Baseline (ES)	DFS (Flat)	DFS (Typed)
MRR	0.47	0.53	**0.55**	0.47	0.53	**0.54**
Hits@1	0.38	0.44	**0.47**	0.39	0.46	**0.47**
Hits@5	0.58	**0.64**	**0.64**	0.55	**0.62**	**0.62**
Hits@10	0.64	0.68	**0.69**	0.60	0.65	**0.66**

Table 2. MRR and Hits@K results for the questions in the private dataset using flat and typed facets.

Metric	Baseline (ES)	DFS (Flat)	DFS (Typed)
MRR	0.29	0.34	**0.39**
Hits@1	0.24	0.28	**0.34**
Hits@5	0.32	0.38	**0.42**
Hits@10	0.34	0.42	**0.46**

Table 3. Sample queries with the dynamically generated flat facets from the first dataset.

Fully specified query	Actual user query	Ranked (by similarity score) dynamically generated (flat) facets
How to configure SSL using mutual authentication in DataPower MQ client?	DataPower MQ client security	**Personal certificate, mutual authentication mode, aix, windows, linux, hpux**, MQ Server, Websphere MQ, **SSL, SSLV3**, ftp server, application server
NMAagent installation failure	NMA failure	**TPC Data Agent, CANDLEHOME directory, configuration failure, GSKit files**, configuration phase, installed agents, **failed probe**, Agent Builder agents, **N4 agent, failed install**
Mismatched MQ jars in my application server	MQ jar mismatch	**Java EE server**, Java Compute Nodes, jar package, Restart SBI, **JVM classpath**, **com.ibm.mq.allclient.jar**, client jar file, MQ WMQ, MQ client jars, **Java EE application server**, dhbcore.jar
How can I export a private key from DataPower Gateway Appliance?	Export key datapower	**OpenSSH format, Personal Certificates section**, key database password, **PFX file, certificate object**, key ring file, stash password, unencrypted file, **pkcs12 certificates**, **tklmKeyImport command**
How to refresh a DataPower WebService Proxy when WSDL got changed?	Refresh webservice proxy	Proxy recorder, **WSDL description, proxy gateway**, DNS Static Host, **Multi - Protocol Gateway, Service Gateway, web service gateway**, **service endpoint**, **MPGW service**

The 3rd column shows the top n dynamically generated facets, ***personal certificate, mutual authentication mode, aix, windows, linux, hpux***, MQ Server, Websphere MQ, **SSL, SSLV3**, ftp server, application server. Facets in **BOLD** were considered useful, as they augmented the actual user query.

We performed similar qualitative evaluation on the private dataset as shown in Table 4. It appears from these random queries that on average 60% facets generated by the system are useful, based on the aforementioned criteria.

4.3 SME User Evaluation

In order to further validate the usefulness of the generated facets with real users. We conducted formal evaluation with 2 SMEs to capture their feedback on the relevance of the flat facets in context of the query. We provided them 50 actual user queries (collected from the production usage logs for the private dataset) and their respective top 20 flat facets. The SMEs' task was to evaluate and provide feedback on the facet relevance in context of the query.

SMEs decided whether the facets are useful based on the query and context that they have based on their skill and expertise. Among the annotations of the two SMEs, there was a Cohen's kappa coefficient of 0.49 inter-annotator agreement, which is a moderate agreement[7]. This is understandable because feedback of the users is subjective and it depends on their skills and experience.

Table 5 shows the average percentage of facets that the SME users found relevant for the given query is about 50% (out of top 20 facets per query). It is important to note that the standard deviation is high, which is indicative of large variance in the quality of the facets for a given query. This has opened additional avenues of investigation to further analyze, predict and understand the cause the under performing queries.

Table 4. Sample queries with the dynamically generated flat facets from the second dataset.

Fully specified query	Actual user query	Ranked (by similarity score) dynamically generated (flat) facets
machine is not powering but there is beep	no power	start up, **HDD**, fan, **AC power**, card, touchpad, **CHKDSK**, beeps, **optical drive, post**
audio is not coming from the speaker my machine	no audio	power, **headphone, music, HDMI, audio output, Bluetooth Headset, dock**, projector, **failed probe**, sound
Help increase the battery life, it keeps on running out fast	increase battery life	power button, **charger, performance, battery pack**, usage, time, **Battery saver**, level, batteries
Card not detected on new system using a specific adaptor	card not detected	**memory**, carrier, power, **adapter, HDD, SIM card**, case, **video card**, **SD Card, SIM**
Dock may fail to connect to a network when USB booting or PXE booting	dock failure	**USB-C, dock driver, docking station, USB**, basic USB, VGA, **Ultra Dock, 3.0 Dock, 3 Dock**

[7] 0.41– 0.60 is considered as moderate [12].

5 Related Work

Given that the goal of our system is to improve document retrieval for user's query, our system recommends facets (some of which are named entities) merely to refine user's problem. With this goal, we briefly highlight research that are closely relevant to our work.

The idea of using the conversational paradigm for information seeking has been originally proposed in [3]. Recently, Radlinski *et al.* [13] propose a theoretical framework for conversational search by suggesting that the interaction based on information retrieval interfaces makes the user experience more natural and convenient. The use of knowledge bases, such as DBpedia [2], has been proven to be effective in measuring the semantic coherence during the dialog sessions [18]. Moreover, structured data models as background knowledge allow the systems to handle complex user queries [16], i.e. retrieving answers which require multi-hop steps. However, the formulation of complex queries requires a considerable cognitive effort for the user. Also, there are certain answers that may not be found with a single query on a given data model. Conversational browsing [17] is proven to be a valuable interaction model in order to satisfy complex users' information needs by iteratively refining the initial query with relevant new concepts. This consideration motivated the idea behind our VA.

Yang *et al.* [19] proposed a learning framework with Deep Matching Networks for response ranking leveraging external knowledge. Knowledge is incorporated using Pseudo-relevance feedback (PRF) (i.e., using top N documents from the initial retrieval of an external collection as feedback to improve the query) and using QA correspondence knowledge distillation. In contrast, in our approach we leverage the knowledge automatically induced from a domain corpus to provide the user dynamically generate facets to refine under-specified queries iteratively.

Table 5. Average and Standard Deviation of flat facet SME Evaluation.

User	Average of relevant facets	Standard Deviation of Relevant Facets
User1	10.90	5.22
User2	10.85	4.84

Aliannejadi et al. [1] describe a conversational search system, which selects clarifying questions for a given query from a large pool of questions. The system uses the BERT [9] model to learn both query and question representations, based on which, it matches questions to a query and conversation context. Different from our approach, it requires a set of manually curated candidate questions.

6 Conclusions and Future Work

In this paper, we proposed a novel and general approach for initializing VAs by leveraging knowledge automatically induced from technical documentations

with the aim to provide a better user experience than a conventional keyword-based retrieval paradigm. Our current system is driven by a dynamic faceted search algorithm. One of the building blocks of our proposed approach is an unsupervised automatic taxonomy induction method.

The evaluation on two real-world datasets in the IT domain has shown the effectiveness of our approach in better refining the users' information need about solving their technical issues. Though it is out of scope for this paper, we have tried out the proposed system in other domains such as medical, oil and gas, legal and the early feedback from the users is positive. The proposed system does not have any domain-specific methods. Thus, it can be easily applied to any domain.

Based on the real-world user evaluation, we learnt that certain user queries do not perform well for facet generation. Though it may be challenging, it will be an interesting exercise to predict the queries that will tend to perform badly.

One of the challenges we identified in the real-world SME evaluation is that it is hard to evaluate the usefulness of the facets automatically as it is highly dependent on the users skill set, domain knowledge and experience. In the SME evaluation, some users annotated some facets as relevant because they had experience of the exact problem in the question whereas as some others did not identify the connection between the problem and the facet.

In our vision, this approach is an early yet fundamental step toward an unsupervised conversational VA for technical support: a system that is able to be easily adapt to new domains, and which does not require any training data.

As future work, we plan to evolve our system by augmenting and strengthening the generation and selection of facets with automatically generated natural languages questions guided by a knowledge graph with a wider range of relation types. Furthermore, at the moment, we are only using taxonomic relations in text for generation and organisation of facets. We plan to induce knowledge graphs with all other relations in the text using techniques such as OpenIE and use them for facet generation.

References

1. Aliannejadi, M., Zamani, H., Crestani, F., Croft, W.B.: Asking clarifying questions in open-domain information-seeking conversations. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval SIGIR 2019, pp. 475–484 (2019)
2. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: a nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC/ISWC -2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76298-0_52
3. Belkin, N.J.: Anomalous states of knowledge as a basis for information retrieval (1980)
4. Castelli, V., et al.: The TechQA Dataset. ArXiv abs/1911.02984 (2019)
5. Chowdhury, M.F.M., Farrell, R.: An efficient approach for super and nested term indexing and retrieval. CoRR abs/1905.09761 (2019). http://arxiv.org/abs/1905.09761

6. Cui, L., Huang, S., Wei, F., Tan, C., Duan, C., Zhou, M.: Superagent: a customer service chatbot for e-commerce websites. In: ACL : System Demonstrations (2017)
7. Dash, D., Rao, J., Megiddo, N., Ailamaki, A., Lohman, G.: Dynamic faceted search for discovery-driven analysis. In: Proceedings of the 17th ACM Conference on Information and Knowledge Management (2008)
8. Dash, S., Chowdhury, M.F.M., Gliozzo, A., Mihindukulasooriya, N., Fauceglia, N.R.: Hypernym detection using strict partial order networks. In: AAAI (2020)
9. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: NAACL 2019 (2019)
10. Fauceglia, N.R., Gliozzo, A., Dash, S., Chowdhury, M.F.M., Mihindukulasooriya, N.: Automatic taxonomy induction and expansion. In: EMNLP : System Demonstrations (2019)
11. Hearst, M.A.: Automatic acquisition of hyponyms from large text corpora. In: COLING (1992)
12. McHugh, M.L.: Interrater reliability: the kappa statistic. Biochemia Med. Biochemia Med. **22**(3), 276–282 (2012)
13. Radlinski, F., Craswell, N.: A theoretical framework for conversational search. In: CHIIR 2017, pp. 117–126. ACM (2017)
14. Safran, N.: Psychology of the searcher, April 2015. https://www.immagic.com/eLibrary/ARCHIVES/GENERAL/BNILE_US/B150428S.pdf
15. Tunkelang, D.: Faceted search. Synth. Lect. Inf. Concepts Retrieval Serv. **1**(1), 1–80 (2009)
16. Usbeck, R., Gusmita, R.H., Ngomo, A.N., Saleem, M.: 9th challenge on question answering over linked data. In: ISWC 2018. CEUR, vol. 2241, pp. 58–64 (2018)
17. Vakulenko, S.: Knowledge-based conversational search. CoRR abs/1912.06859 (2019). http://arxiv.org/abs/1912.06859
18. Vakulenko, S., de Rijke, M., Cochez, M., Savenkov, V., Polleres, A.: Measuring semantic coherence of a conversation. In: Vrandečić, D., et al. (eds.) ISWC 2018. LNCS, vol. 11136, pp. 634–651. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00671-6_37
19. Yang, L., et al.: Response ranking with deep matching networks and external knowledge in information-seeking conversation systems. In: SIGIR 2018, pp. 245–254 (2018)