PreFace: Faceted Retrieval of Prerequisites Using Domain-Specific Knowledge Bases

Prajna Upadhyay(✉) and Maya Ramanath

IIT Delhi, Hauz Khas, New Delhi 110016, India
{prajna.upadhyay,ramanath}@cse.iitd.ac.in

Abstract. While learning new technical material, a user faces difficulty encountering new concepts for which she does not have the necessary prerequisite knowledge. Determining the right set of prerequisites is challenging because it involves multiple searches on the web. Although a number of techniques have been proposed to retrieve prerequisites, none of them consider grouping prerequisites into interesting facets. To address this issue, we have developed a system called **PreFace** that (i) automatically determines interesting facets for a given concept of interest, and, (ii) determines prerequisites for the concept and facet. The key component of PreFace is a retrieval model that balances the trade-off between the relevance of the facets and their diversity. We achieve this by representing each facet as a language model estimated using a domain-specific knowledge base and a large corpus of research papers, and ranking them using a risk-minimization framework. Our evaluation of the results over a benchmark set of queries shows that PreFace retrieves better facets and prerequisites than state-of-the-art facet extraction techniques.

Keywords: Facets · Prerequisite · Knowledge base

1 Introduction

When reading new technical material, a common problem faced by readers is encountering new and unknown concepts, to understand which the reader does not have the required prerequisite knowledge. A prerequisite for a concept `a` is another concept `b` that can be suggested for study before `a` for better understanding of `a`. For example, to understand `artificial_neural_network`, one needs to have a prerequisite knowledge of `perceptron` and `activation_function`. Also, a knowledge of a programming language, such as `matlab`, will help the user in implementing `artificial_neural_network`, and a knowledge of `phoneme` will help her understand applications of `artificial_neural_network`, because `phoneme_recognition` is an application of `artificial_neural_network`. So, a prerequisite also includes concepts that help the user understand the multiple factes of a queried concept. Identifying the right set of prerequisite concepts is challenging because retrieval systems only return relevant documents that may or

J. Z. Pan et al. (Eds.): ISWC 2020, LNCS 12506, pp. 601–618, 2020.
https://doi.org/10.1007/978-3-030-62419-4_34

Table 1. Example of 4 facets and prerequisites retrieved by **PreFace** for `artificial_neural_network`

Facets
`neural_network`, `perceptron`
`binary_classification`, `medical_classification`, `phoneme`
`backpropagation`, `optimization_algorithm`, `gradient_descent`,
`octave`, `matlab`

may not contain prerequisites in them. Even if they do, the user may need to further refer to the prerequisite's prerequisite. This leads to a chain of searches and is time-consuming for the user. It would be helpful to have a retrieval system that, given an input concept, returns exactly the prerequisite concepts required to understand it. A number of techniques to determine prerequisites for a concept have been proposed over the years. Most of them address this problem by constructing prerequisite functions that take in a pair of concepts and determine whether one is a prerequisite of the other [21,22,28] or by constructing prerequisite graphs [39], [31,41]. These techniques have typically relied on features from textual sources, structure of textbooks and learning from training examples to construct or learn prerequisite relationships.

The main issue with these techniques is that they ignore the multiple facets of understanding of a query. Consequently, the prerequisites returned by these techniques are not grouped into facets. For example, for `artificial_neural_network`, along with concepts such as `perceptron`, concepts such as `matlab`, `octave` or `phoneme` are returned together. `matlab` and `octave` are relevant if the user is interested in implementing `artificial_neural_network`. So, instead of returning them together, it is desirable if they are returned as facets. Table 1 shows four facets for prerequisites for `artificial_neural_network`, 1) `neural_network` and `perceptron`, which are related to `neural_networks` 2) `binary_classification`, `medical_classification`, and `phoneme`, which help the user understand *applications* of `artificial_neural_network`, 3) `backpropagation`, `optimization_algorithm`, and `gradient_descent`, which help the user understand *algorithms* related to the query, and 4) `octave` and `matlab`, which help the user implement the query. All four of them are important towards an overall understanding of the concept. So, instead of returning all of them together, it is desirable to have a system that returns them as facets or groups.

One way to build such a system is to use existing techniques to solve the two sub-problems separately, namely facet extraction [9,18,32] and prerequisite determination [4,22] i.e. we can first extract facets for a query and then only retain concepts that are prerequisites to the query in each facet. However, this approach does not guarantee good results. There are two main reasons for this. Firstly, query-based facets are generally extracted using entities and relationships from open-domain knowledge bases like Freebase [18]. Using Freebase to retrieve facets for queries in Computer Science retrieves little or no results

because the domain-specific relationships and entities are under-represented in them [37]. So, there is a need to use domain-specific knowledge bases such as TeKnowbase [36,37]. The second reason is that the proposed techniques assume that the knowledge base is sufficient to extract facets from FreeBase [9,18]. The key challenge they address is the efficient ranking of facets because they generate a large number of candidate facets in the open domain. This assumption does not hold for domain specific knowledge bases such as TeKnowbase because they are sparse. Moreover, these techniques retrieve poor quality facets when the knowledge base is from a different domain. We need a new technique to automatically extract more and better quality facets.

To address these issues, we introduce the novel problem of faceted prerequisite extraction and develop **PreFace**[1], which solves the problem of facet extraction and prerequisite determination for a query. We formulate this as a retrieval problem where we first generate high quality facets using TeKnowbase and the Open Research Corpus [1]. The facets and the query are then represented as language models [29], and ranked balancing the trade-off between the relevance and diversity of retrieved facets. Our evaluation over a standard benchmark set of queries shows that PreFace extracts better facets and prerequisites than state-of-the-art facet and pre-requisite extraction systems.

Contributions. The salient contributions of this paper are:

- Introduction of the novel problem of faceted retrieval of prerequisites
- Development of PreFace, which is a language model framework to retrieve interesting facets as well as prerequisites for a query of interest using a domain-specific knowledge base and a corpus of research papers.
- Demonstrating that PreFace can retrieve better facets and prerequisites together than separately retrieving them using state-of-the art techniques.
- PreFace is designed to work with *any* prerequisite function.

This paper is organized as follows. Section 2 describes the retrieval model for PreFace and the estimation techniques. Section 3 describes the experimental setup and results are discussed in Sect. 4. Section 5 discusses the related work.

2 Framework for PreFace

Figure 1 shows the architecture of PreFace. It takes a query and returns a ranked list of facets containing prerequisites. The terminologies are as follows:

Concept. A concept is any technical topic that can be studied and understood. In our case, a concept is an entity in the domain of Computer Science.

Query. A query is a concept about which the user wishes to study. For example `artificial_neural_network` is a query.

[1] The code and dataset for PreFace is available at http://www.cse.iitd.ac.in/~prajna/preface.html.

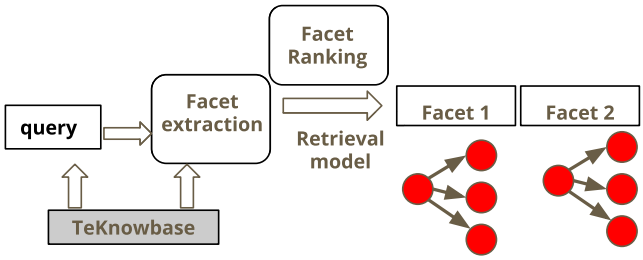

Fig. 1. Components of PreFace. It takes a query and returns a ranked list of facets containing pre-requisites for the query.

Aspect. An aspect describes some subtopic of the query that the user is interested in. For example, *application* or *implementation* are aspects of interest for `artificial_neural_network`.

Prerequisite. A prerequisite of a concept `a` is another concept `b` that can be suggested to be studied before `a` for better understanding of `a`. For example, `perceptron` is a prerequisite for `artificial_neural_network`. It also includes concepts such as `matlab` or `phoneme`, which helps the user understand different aspects of the query, such as *implementation* or *application* respectively.

Facet. A facet is a set of prerequisites to the query that describe an aspect of the query. For example, facet number 4 in Table 1 consists of `matlab` and `octave` which are pre-requisites for the *implementation* aspect for `artificial_neural_network`.

An overview of the main components of PreFace is described as follows:

Facet Extraction. This component generates candidate facets using TeKnowbase and a corpus of research papers.

Ranking of Facets. The extracted facets as well as the query are represented as language models. The language model for the query is estimated using TeKnowbase. The candidate facets are ranked balancing the trade-off between the similarity of their language model with the query language model, and the diversity of the retrieved facets.

TeKnowbase. TeKnowbase is the backbone of PreFace. It helps in the extraction as well as the modeling of the relevance of the facets. All these components are described in details in the following subsection.

2.1 Facet Extraction

A facet is a group of prerequisites for the query that describes some aspect of the query. We used the relevant documents returned in the top positions for the query to generate candidate facets. The key idea used is to extract frequently occurring key-phrases from these documents and cluster them to generate facets. Below we describe in detail these two steps:

Extracting Key-Phrases from Documents. We indexed the Open Research Corpus dataset[2] on Galago[3] and retrieved top-1000 documents for a query. We then used Rake[4], a tool to extract and score essential phrases from a document. To further clean the list of extracted phrases, we performed two data cleaning operations as follows – i) we retained only those phrases that had a length of at most 5, and ii) phrases with a score (returned by Rake) less than 5 were removed from the set of candidate phrases. The main reason for using phrases instead of entities or terms is because phrases capture the context better than entities. For example, consider the phrase `applications like robot navigation` and `robots using new camera technologies` retrieved for query `computer vision`. The phrases mention entities from TeKnowbase such as `robot navigation` or `camera`. Additionally, presence of terms, which are not entities in TeKnowbase, such as `applications` and `using` indicate that these phrases are relevant for the *application* aspect for `computer vision`, because they are modeled by our aspect-based retrieval model [38].

Clustering Key-Phrases into Facets. The next step is to cluster these phrases and obtain candidate facets. To generate semantically related clusters, we used a bag of entities representation for the key-phrases along with bag of words. The phrases were tagged with entities from TeKnowbase. Additionally, this set was expanded by adding entities situated at a 1-hop distance from already tagged entities in TeKnowbase. This was done to capture better context. Consider the phrase `using backpropagation` extracted for `artificial neural network`. The triple ⟨`backpropagation, type, algorithm`⟩ exists in TeKnowbase, so `algorithm` exists in its 1-hop neighborhood and will be added to its feature set. So, it will have high similarity to other phrases containing `algorithm`, and will likely appear in the same cluster as them. Having such a representation, these phrases were clustered into semantically related groups using agglomerative clustering algorithm with complete linkage. The distance between the phrases was measured using cosine distance.

2.2 Ranking of Facets

Modelling Facet Relevance. Given a query q, we address this problem by defining a language model for the query as well as the facet. As already defined in Sect. 2, a facet consists of prerequisites relevant for some aspect of the query. So, it should contain terms that are found in prerequisites of the query, as well as terms relevant for the query and the aspect. The challenge lies in identifying the pre-requisites and the aspects for the query, and then modeling the relevance for the query and the aspect. The facets should also contain items that are highly similar to each other. So, the relevance is modeled by two components:

[2] https://allenai.org/data/s2orc.
[3] https://www.lemurproject.org/galago.php.
[4] https://pypi.org/project/rake-nltk/.

1) Query dependant facet relevance. This is denoted by $PreFace(w|q)$.

$$PreFace(w|q) = \lambda Preq(w|q) + (1-\lambda)\sum_i P(q_i|q)P(w|q,q_i), \tag{1}$$

It consists of two components, $Preq(w|q)$ and $\sum_i P(q_i|q)P(w|q,q_i)$, which are mixed using λ. These two components are described as follows:

i) Prerequisite probability. $Preq(w|q)$ models the probability of a word w appearing in a prerequisite for the query.
ii) Query and aspect probability. To be able to retrieve other concepts that help understand other aspects of the query, we first have to identify these aspects.

After identifying the set of aspects A with $q_i \in A$, the relevance of a term w for q and q_i is modeled using the technique proposed by us in [38]. The query and the aspect component is represented by $P(w|q,q_i)$ and is modeled as follows:

$$P(w|q,q_i) = \gamma P_{ind}(w|q_i) + (1-\gamma)P_{dep}(w|q,q_i) \tag{2}$$

where $P_{ind}(w|q_i)$ is the component determined by the aspect alone and $P_{dep}(w|q,q_i)$ is the query dependent component, determined by both the query and the aspect. γ is used to mix these two components.

2) Query independent facet relevance. The query independent facet relevance models the relevance of the facet independent of the query. This is determined by the quality of facet, dependent on the size of the facet and the strength of the similarity between its items. A facet that contains a large number of similar entities is more important than one that contains lesser number of entities, provided the strength of similarity between the two is the same. So, we first apply a similarity threshold on the facets, and then use the size of the facet as a measure of its quality. The query independent facet relevance is denoted by $Q(f)$.

Algorithm 1: Probabilistic framework for PreFace (λ, τ, F, q)

$S = \emptyset$;
$PreFace(w|q) = \lambda Preq(w|q) + (1-\lambda)\sum_i P(q_i|q)P(w|q,q_i)$;
$f* = \text{argmax}_{f \in F} Q(f) * \frac{1}{D_{KL}(PreFace||M_f)}$;
$S = S \cup \{f*\}$;
while $|S| < \tau - 1$ **do**
  $f* = \text{argmax}_{f \in F \setminus S} Q(f) * \frac{D_{KL}(M_f||M_S)}{D_{KL}(PreFace||M_f)}$;
  $F = F \setminus \{f*\}$;
  $S = S \cup \{f*\}$;
end
return S

Modeling Facet Diversity. Algorithm 1 describes the procedure for the ranking of facets. It takes a query (q), the number of facets to be returned (τ), the mixing parameter to combine the two components of Eq. 1 (λ), and the set of candidate facets (F) as input and returns a ranked list of facets S. It greedily selects the best facet at each iteration that balances the trade-off between its relevance to the query and dissimilarity to the facets already retrieved. The user should be recommended facets that are diverse to one another. This means that if a facet describing the algorithms for understanding `artificial_neural_network` appears in the top-k position, it is desirable that it is not suggested again. This is done by returning facets whose language model diverges the most from the language model of already retrieved facets and least from the query. Each facet is scored according to the following equation:

$$\frac{D_{KL}(M_f||M_S)}{D_{KL}(PreFace||M_f)} = \frac{\sum_w M_f(w) log \frac{M_f(w)}{M_s(w)}}{\sum_w PreFace(w) log \frac{PreFace(w)}{M_f(w)}} \tag{3}$$

where M_f is the language model of a facet f and M_S is the language model representation of the set of facets already retrieved. $D_{KL}(M_f||M_S)$ is the KL divergence between facet language model and the language model for the set of facets already retrieved. $D_{KL}(PreFace||M_f)$ is the KL divergence between the language model for the query ($PreFace(q)$) and the language model for the facet. The language model of a facet f and set S is described by the following equations. These models are smoothed using additive smoothing techniques.

$$M_f(w) = \frac{tf(w,f)+1}{length(f)+|V|} \quad (4) \qquad M_S(w) = \frac{\sum_{f_i \in S} tf(w,f_i)+1}{\sum_{f_i \in S} length(f_i)+|V|} \quad (5)$$

where $tf(w,f)$ and $length(f)$ is the frequency of w in f and number of terms in f, respectively. In other words, greater the divergence between the language model of the facet and language model of S, better is the facet and lower the divergence between the query language model and the facet language model, better the facet. The facets are re-ranked at each iteration, until the desirable number (τ) of facets are retrieved using the following equation:

$$f* = \text{argmax}_{f \in F \setminus S} Q(f) * \frac{D_{KL}(M_f||M_S)}{D_{KL}(PreFace||M_f)} \tag{6}$$

where $Q(f)$ is the quality of a facet. The facet retrieved at this position is removed from the pool of candidate facets and added to S.

Estimation of Components. In this section, we describe the techniques to estimate the probabilities that are used in our probabilistic framework.

Estimating $Preq(w|q)$. This component models the probability of words likely to appear in prerequisites of the query. To estimate this component, we can use any standard prerequisite function from the literature. For our work, we used the well-known prerequisite function, $RefD$ (or reference distance) [22], because it is a simple, unsupervised metric and gives good results. $RefD$ takes two

concepts as input and returns a score denoting the strength of the pre-requisite relationship between them. We further identified the set of siblings $Siblings(q)$ of q (entities in TeKnowbase that share same parent in its taxonomy) and also their pre-requisites to estimate $Preq(w|q)$. At first, we computed $Preq_s(w|s_i)$ for each sibling s_i of q in $Siblings(q)$.

$$Preq_s(w|s_i) = \frac{\sum_{e_k \in ent(w)} RefD(e_k, s_i)}{\sum_{e_k \in C} RefD(e_k, s_i)}, RefD(e_k, s_i) > thresh \tag{7}$$

Then, $Preq(w|q)$ is estimated as follows:

$$Preq(w|q) = \frac{1}{|Siblings(q)|} \sum_{s_i \in Siblings(q)} Preq_s(w|s_i) \tag{8}$$

where C is the concept space and $ent(w) \subset C$ is the set of entities that contain the term w in TeKnowbase. The intuition behind using a mixture of $Preq_s(w|s_i)$, where s_i is an entity that shares the same parent as q is that siblings in TeKnowbase taxonomy have similar prerequisites, and we can make our distribution more accurate by including concepts that were not returned as prerequisites for q.

Estimating Query and Aspect Probability. The second component in Eq. 1 models the query and the aspect probability. To estimate this component, we first have to identify the set A of aspects for q. As already described in Sect. 2, aspects describe some subtopic for the query, which can be entities or relationships in TeKnowbase. The key-idea used is to acquire entities from key-phrases and cluster them into groups based on the graph structure of TeKnowbase and identify a representative from each group to be used as the aspect. The procedure to generate A is described as follows:

1) Acquiring set of entities. We tagged entities from TeKnowbase in the set of key-phrases retrieved using the procedure described in Sect. 2.1 and obtained the set of relevant entities E. Each $e \in E$ was scored using the following formula:

$$score(e) = co\_occ(e, q) RefD(e, q), RefD(e, q) > thresh \tag{9}$$

where $co\_occ(e, q)$ counts the number of times e and q have appeared together in a document across the top-1000 relevant documents retrieved for q. An entity that frequently co-occurs with the query in relevant documents should be more important to the query, so is assigned a higher score using this formula.

2) Using TeKnowbase entities as aspects. Having a set of entities E, we have to partition it into groups such that each group is highly relevant to the query as well as is semantically similar to each other. We can then choose a representative from each group as an aspect for q. To do this, we used the links in TeKnowbase and created an induced sub-graph G on TeKnowbase using the set of entities in E and then applied the star clustering algorithm [2] on G to cluster entities into groups. Star Clustering algorithm works on a graph and takes a similarity threshold σ as input. It retains edges that have strength greater than

σ and then clusters the nodes of the graph into groups. It is a greedy algorithm that chooses the node with the highest degree first and assigns all its adjacent nodes to its cluster. It repeats the procedure with other nodes that have not yet been assigned to any cluster. In TeKnowbase, all links are assumed to have a strength of 1, so σ was set to 1. We scored each cluster using the sum of scores of its entities, given by: $\sum_{c_i \in C} score(c_i)$, where $score(c_i)$ is the score of each entity calculated using Eq. 9. From every top-10 scored clusters, we chose the highest scored entity to be added to A.

3) Using TeKnowbase relationships as aspects. Apart from using entities in TeKnowbase as aspects, we expanded the set by adding relationships from TeKnowbase as aspects. We used the aspects that were used in [38] for query q, namely *algorithm*, *application* and *implementation*. We also added *type* and *technique* to the set of aspects A.

After generating the set of aspects A, we have to estimate $P(q, q_i)$ for each $q_i \in A$. As already described in Sect. 2.2, we used our previously proposed aspect based retrieval [38] technique to model this component. The query independent and dependent components of this model were estimated as follows:

1) Estimating $P_{ind}(w|q_i)$. We estimated the query-independent aspect probability by explicitly querying for the aspect term alone and retrieving the top documents. To further make our estimation accurate, we retained only those documents that mention the aspect term in the title or abstract. Having this set of documents D, we estimated the query independent aspect probability as follows:

$$P_{ind}(w|q_i) = \frac{1}{|D|} \sum_{d \in D} \frac{tf(w,d)}{\sum_{w' \in d} tf(w',d)} \tag{10}$$

where $tf(w,d)$ is the frequency of w in a document d.

2) Estimating $P_{dep}(w|q,q_i)$. We estimated this component from both the search results and TeKnowbase. To estimate this component from search results, we explicitly queried for q and q_i together and retrieved top ranked documents. To further make our estimation accurate, we only retained those documents that contained both the query and aspect terms in either the title or the abstract. This probability was estimated as follows:

$$P_{docs}(w|q,q_i) = \frac{1}{|D'|} \sum_{d \in D'} \frac{tf(w,d)}{\sum_{w' \in d} tf(w',d)} \tag{11}$$

where $tf(w,d)$ is the frequency of w in a document d and D' is the set of documents with each document containing the query and the aspect term both. We also used TeKnowbase to further improve the accuracy of estimation. TeKnowbase consists of entities connected to q via relation described by the aspect q_i. So, the words appearing in entities connected to the q via q_i should also be considered for estimation. For example, TeKnowbase consists of the triple ⟨`activity_recognition, application, hidden_markov_model`⟩, which implies that `activity_recognition` can be suggested as a prerequisite for the application

aspect. However, TeKnowbase is sparse and only using those links to estimate this probability will not result in an accurate estimation. So, we used the approach adopted in [38] for improving the estimation using meta-paths [34] in TeKnowbase, which computes the probability $DI_{e_i}(e_j)$ of two entities being connected via the relation described by the aspect q_i. This is used to estimate $P_{KB}(w|q, q_i)$ as follows:

$$P_{KB}(w|q, q_i) = \sum_{e \in ent(w)} DI_e(q) \tag{12}$$

We then computed the query and aspect probability by mixing the probability distributions given by Eq. 11 and Eq. 12. The final probability for query and aspect, both, is given as follows:

$$P_{dep}(w|q, q_i) = \alpha P_{docs}(w|q, q_i) + (1 - \alpha) P_{KB}(w|q, q_i) \tag{13}$$

3) Estimating $P(q_i|q)$. This models how important is the aspect q_i to q. All the aspects can be given equal probability or the probability can be proportional to $score(q_i)$, described by Eq. 9. Given A, the set of aspects, the second component of the final relevance equation is a linear combination of the query and aspect probabilities, described as follows:

$$\sum_i P(q_i|q) P(w|q, q_i) \tag{14}$$

The final relevance equation is a mixture of Eq. 14 and Eq. 8.

4) Ranking of facets. After estimating the query dependent probability of terms given a query, we rank the facets according to Algorithm 1.

Item Ranking. After the facets have been ranked, we have to rank the entities in the facet to be shown as pre-requisites to the user. To do so, we first identified a representative element re for each facet. We tagged entities in the facet and chose the most frequently occurring entity as the representative. After choosing the representative, we computed the score for each entity e tagged in the facet as follows: $freq(e)sim(e, re)$, where $sim(e, re)$ is the normalized cosine similarity between the vector representations of entities e and re. These vector representations were obtained by training Node2Vec algorithm on TeKnowbase. The entities in each facet are then ranked in decreasing order of this score.

3 Experiments

3.1 Setup

We experimented with the Wikipedia and the Open Research Corpus datasets. The values of λ, γ and α were set to 0.5. $thresh$ was set to 0. For every baseline, we retrieved the top-5 facets for each query. Then, in each facet, the top 3 items were shown to the users for evaluation.

artificial neural network , backpropagation, collaborative filtering, computer vision, conditional random field, context-sensitive grammar, cross entropy, dimensionality reduction, generative adversarial networks, genetic algorithm, gradient descent, hidden markov model, latent dirichlet allocation, linear regression, logistic regression, optical character recognition, pagerank, probabilistic context-free grammar, question answering, recursive neural network, regular expression, reinforcement learning, sentiment analysis, shallow parsing, singular value decomposition, spectral clustering, speech recognition, statistical machine translation, word-sense disambiguation, word2vec

Fig. 2. Benchmark queries

3.2 Benchmark Queries

We chose 30 queries from the set of topics released by [21] (listed in Fig. 2), which is a set of concepts annotated with their prerequisites. We used these annotations to measure the precision of prerequisites retrieved by our technique as well as the baselines. We additionally conducted user-studies to evaluate the precision of concepts that were retrieved but not already in that dataset. We restricted ourselves to queries that have a Wikipedia page because the prerequisite function $RefD$ uses Wikipedia to compute reference distance between concepts.

3.3 Baselines

As already stated in Sect. 1, we can solve the two sub-problems separately using existing state-of-the-art techniques, as in the baselines mentioned below:

1) QDMKB + RefD. QDMKB [18] is a state-of-the art facet retrieval technique for extracting facets from knowledge bases and search results. It improves upon the results of its predecessor, QDMiner [8] and other state-of-the-art techniques QF-I and QF-J [19]. QDMKB extracts first and second-hop properties (or relationships the query participates in) for the query from the knowledge base, assuming that each property is a candidate facet. It ranks them according to the frequency of the entities appearing in the relevant documents. $RefD$ is a state-of-the-art prerequisite function already described in Sect. 2.2. We implemented QDMKB and extracted facets for a query from TeKnowbase. Then, for each facet, we retained only those entities that were returned as prerequisites for the query according to $RefD$.

2) RefD + TKB. We can also retrieve facets for prerequisites of a query by clustering them into groups. We used the links in TeKnowbase to cluster the candidate prerequisites into meaningful groups. We tagged entities from TeKnowbase in the top-1000 results. We then constructed an induced sub-graph from these entities on TeKnowbase and applied star clustering algorithm [2] on it, as described in Sect. 2.2. Each cluster was scored according to the following equation. The clusters were then ranked according to $score(C)$ (Eq. 15).

$$score(C) = \sum_{c_i \in C} RefD(c_i, q), RefD(c_i, q) > thresh \tag{15}$$

3) PreFace. This is our technique that extractes facets using TeKnowbase and a corpus of research papers and then ranks them by representing them as language models using Algorithm 1.

3.4 Evaluation Scheme

There are two components to evaluate – 1) quality of facets 2) quality of prerequisites. Owing to the lack of a standard dataset for evaluation, we used human evaluators (computer science researchers) to evaluate the generated facets. Top 5 facets with 3 items in each facet were shown to two evaluators (computer science researchers) who evaluated it for its quality.

Evaluation of Facet Quality. To evaluate the quality of facets retrieved, we conducted the following experiments. Each facet was evaluated for the quality of clustering and ranking of facets. We used semantic similarity to measure the quality of clusters retrieved by each technique and DCG (Discounted Cumulative Gain) to measure the ranking of facets. The methodology to evaluate both the qualities is described as follows:

Ranking of Facets. To evaluate the ranking quality, each user was shown a representative item from each facet and asked if that item was relevant to the query. For Preface, the entity that was most frequently occurring in the facet was shown as the representative. For RefD + TKB, the representative was the star centre of each cluster generated by the star clustering algorithm. For QDMKB + RefD, we chose the item in the facet that obtained the highest score for RefD. The relevance scores could be 0: not relevant at all, 1: somewhat relevant, and 2: very relevant. We then used these scores to compute DCG values.

Clustering Quality. We have to evaluate the quality of facets generated by our technique as well as the baselines. To judge the clustering quality, we asked the user to score the similarity between other entities in the facet to the representative of the facet. The score could be 2: very similar, 1: somewhat similar, and 0: not similar at all. For example, `quadratic_convex_function` and `general_smooth_nonlinear_function_approximator` are similar to each other because both are functions, whereas `algorithm` and `integer` are not similar to each other. For PreFace, we showed the top-3 items ranked according to the approach described in Sect. 2.2. For RefD + TKB, we showed the top-3 entities scored according to $RefD$ scores that were added to the clusters after their star centres were chosen. For QDMKB, top 3 entities ranked in decreasing order of their $RefD$ scores were shown. We computed the score for each pair and normalised the similarity score so that it lies between 0 and 1.

Evaluation of Prerequisites. Apart from evaluating the quality of facets, we also have to evaluate the prerequisites retrieved by our technique. We used the set of prerequisite concept pairs released by [21] as the ground truth. The same items that were shown to the user for evaluating the quality of facets were shown to be evaluated as prerequisites. Since the definition of our prerequisites is broader than in earlier work, we performed a user study to judge the relevance of prerequisites not present in the dataset. The top 5 facets were used for evaluation of prerequisite concept. Each prerequisite was annotated by 2 evaluators (computer science researchers) with scores of 0 or 1. A score of 1 was assigned if the prerequisite was judged to be relevant for the aspect described by the representative item of that facet. In other cases, a score of 0 is assigned.

4 Results and Discussion

4.1 Results

Facet Quality. Table 2(a) shows the values for facet quality for all 3 techniques. Our technique outperforms the baselines in retrieving better quality facets.

Table 2. Tables showing results for a) DCG and cluster similarity values for facet ranking and facet quality, respectively, for all 3 techniques b) Precision of prerequisites retrieved by PreFace and competing techniques.

Techniques	DCG @5	Cluster similarity
PreFace	**5.80**	**0.95**
QDMKB + RefD	4.26	0.80
RefD + TKB	5.56	0.77

Techniques	Precision
PreFace	**0.76**
QDMKB + RefD	0.636
RefD + TKB	0.68

Clustering Quality. The average cluster similarity for our technique was 0.95, which is very high and the highest among other techniques. This was possible because of the bag-of-words and entities representation of items using TeKnowbase. QDMKB + RefD obtains a cluster similarity score of 0.8. The reason for it performing worse than PreFace is that not all the facets returned by QDMKB + RefD contain semantically similar items with respect to the query. For example, Table 3 shows the facets retrieved for `artificial_neural_network` by all the techniques. QDMKB + RefD returns a facet consisting of items `machine_learning`, `probability`, `hidden_markov_models` and `decision_tree`. This facet was returned because all the items in the facet are related to `artificial_neural_network` via the sequence ⟨`research(relatedTo), artificial_intelligence, techniquein_inverse`⟩. Amongst these items, `probability` is not semantically similar to `machine_learning`. So, all the second hop properties may not lead to good quality facets. RefD + TKB obtains a cluster similarity score of 0.77. The reason for the lower

similarity value is that the semantics of relations are not considered while clustering the entities. For example, for the same query, a facet retrieved at 3rd position consists of `computer_science` and `matlab`. Both these items are not similar to each other, but have appeared together because they participate in the following triples with `programming_language`: ⟨`matlab, typeof, programming_language`⟩ and ⟨`programming_language, issoftware_notations_and_tools(relatedto), computer_science`⟩ in TeKnowbase.

Ranking Quality. The ranking quality of the facets is measured using DCG. Our technique outperforms the baselines in ranking of facets. QDMKB + RefD performs worse because it fails to retrieve at least 5 facets for all the queries. RefD + TKB performs better than QDMKB + RefD. Overall, PreFace generates much better facets as compared to its competitors.

Quality of Prerequisites. Table 2(b) shows the precision of retrieved prerequisites. PreFace outperforms both the baselines by obtaining a precision of 0.76 across all queries. RefD + TKB comes second in retrieving prerequisites to our technique because it uses $RefD$ to construct facets. It obtains a precision of 0.68. QDMKB + RefD performs the worst because it returns few prerequisites in each facet and the facet is also not relevant to the query. This shows that our retrieval system is able to return better prerequisites than other techniques.

Table 3. Prerequisites retrieved for `artificial_neural_network` by PreFace, QDMKB + RefD and RefD + TKB for top 6 facets

PreFace	**RefD + TKB**	**QDMKB + RefD**
`neural_network`, `network_model`, `perceptron`, `backpropagation`	`integer`, `turing_machine`, `information_theory`, `differential_equation`	`software`, `theorem`
`genetic_algorithm`, `backpropagation`, `optimization_algorithm`, `gradient_descent`	`mathematics`, `statistics`, `regression_analysis`, `statistical_inference`	`machine_learning`, `probability`, `hidden_markov_models`, `decision_tree`
`transfer_function`, `activation_function`, `linear`, `basis_function`	`computer_science`, `matlab`, `fortran`, `natural`	`metaheuristic`, `syntactic_pattern_recognition`
`computer_security`, `program`, `laptop_computer`, `quantum_computer`	`continuous_function`, `transfer_function`, `computable_function`, `integral`	
`binary_classification`, `medical_classification`, `tumor`, `phoneme`	`probability_distribution`, `multiplication`, `linear_system`, `impulse_response`	
`statistical_software`, `system_software`, `octave`, `matlab`	`machine_learning`, `artificial_intelligence`, `statistical_estimation`	

5 Related Work

While ours is the first attempt at retrieving faceted pre-requisites, both of these techniques i.e., facet extraction and prerequisite determination have been independently explored. Below we review both the approaches.

5.1 Facet Extraction

Extracting facets has been studied over a long time, using knowledge graphs and/or search results. Below we list related work in each of these areas.

Facets are either pre-defined categories on the corpus or are built dynamically based on the query. Among existing work that use static facet categories are Ontogator [26] and mSpace [32] that use RDF graphs to annotate images to facilitate faceted browsing. [27] developed BrowseRDF, that helps the user browse an RDF graph by providing constraints to be applied on graph properties. They also proposed metrics to measure the quality of facets and rank them. [14] helps a user answer complex queries using faceted search on Wikipedia. The properties are extracted from Wikipedia info-boxes and displayed to the user for further refining the results. gFacet [16] and VisiNav [15] are tools that provide visualization of web of data supported with faceted filtering techniques using RDF graphs and properties. [6] proposed a system to construct facet hierarchies for a text corpus and then assign the documents to each of these facets. This is different from our facets which are co-ordinate and not hierarchies.

Among the systems that generate facets dynamically are those that build SPARQL queries on the fly to be executed on the respective SPARQL endpoints. [10] and [9] have used these approaches to build facets for a query. The authors proposed QDMiner [8] to retrieve facets from search results by extracting frequently occurring lists in relevant documents. These lists were extracted from structured data in the documents, like HTML lists or tables. [18] proposed QDMKB that improved the results generated by [8] by using FreeBase. Another extension to QDMiner was done by [19] where they improved the quality of facets generated by using a probabilistic graphical model. Both QDMiner and QDMKB assume that the corpus is rich in meta-data, which is not always true. The techniques that use knowledge bases to generate facets assume that they are sufficient, which may not be the case for domain-specific graphs.

In the context of academic search, [11] built *Scienstein* that allows users to search for papers using authors or reference lists apart from the usual keyword-query search. These tools make use of the underlying structure of the citation graph as well as machine learning techniques on the document text. [7] made an effort to provide faceted retrieval for research papers in computer science. The facets were – publication years, authors or conferences. These facets are different from aspects in our scenario. [3] proposed techniques to further categorise the relationships between the query paper and the recommended paper. These relationships are expressed in the form of facets like `background`, `alternative_approaches`, `methods` and `comparisons`. [5] used similar facets to

summarize scientific papers. These facets are extracted by identifying the context of the text surrounding the citation.

5.2 Prerequisite Determination

Among the techniques that determine prerequisites between a pair of concepts, [35] used crowd-sourcing to create a gold standard dataset that was used to train a classifier using features from Wikipedia. In [22], the authors proposed $RefD$ using frame semantics to compute prerequisites between concepts using Wikipedia. In [31], the authors make use of Wikipedia clickstream data to build the classifier. In [23], the authors used $RefD$ to infer concept prerequisite relationships from course prerequisite pairs. They later proposed active learning techniques to reduce the amount of training data in [24]. [12] proposed information theoretic measures to determine dependencies between concepts in a scientific corpus. In [28], authors trained classifier using lecture transcripts from MOOCs to improve the prediction of prerequisite pairs. [41] proposed an optimization framework to model prerequisite links among concepts as latent links which can be used to infer prerequisite links between course pairs across universities. [30] proposed a neural model using siamese networks to improve prediction of prerequisite relations. Supervised learning model has been used in [25] to determine prerequisite relations by extracting high quality phrases from educational data.

A number of techniques have been proposed to present information in an organized manner. These include generating hierarchies over document collections [20] or ordering documents in a sequence. [33] proposed metro-maps to show the developments between research papers. [40] proposed methods to generate reading orders for a concept of interest in the domain of physics. [13] and [17] proposed techniques to generate reading lists of research papers for a query.

Although a number of techniques exist that solve the two sub-problems independently, to the best of our knowledge, there exists no other system that solves both of these problems together.

6 Conclusion

In this paper, we developed PreFace, a system to automatically extract facets together with prerequisites for a concept of interest. To the best of our knowledge, ours is the first system that solves this problem. PreFace extracts facets using TeKnowbase and a corpus of research papers and represents them as language models. It then ranks them by balancing the trade-off between their relevance and diversity. Our evaluation of the results shows that PreFace retrieves better facets and prerequisites than state-of-the art techniques.

References

1. Ammar, W., et al.: Construction of the literature graph in semantic scholar. In: NAACL (2018)

2. Aslam, J.A., Pelekhov, E., Rus, D.: The star clustering algorithm for static and dynamic information organization. J. Graph Algorithms Appl. **8**(1), 95–129 (2004)
3. Chakraborty, T., et al.: FeRoSa: a faceted recommendation system for scientific articles. In: PAKDD (2016)
4. Chen, Y., Wuillemin, P.H., Labat, J.M.: Discovering prerequisite structure of skills through probabilistic association rules mining. In: EDM (2015)
5. Cohan, A., Goharian, N.: Scientific document summarization via citation contextualization and scientific discourse. Int. J. Digit. Libr. **19**(2–3), 287–303 (2018)
6. Dakka, W., Ipeirotis, P.G.: Automatic extraction of useful facet hierarchies from text databases. In: ICDE (2008)
7. Diederich, J., et al.: Demonstrating the semantic growBag: automatically creating topic facets for FacetedDBLP. In: JCDL (2007)
8. Dou, Z., et al.: Finding dimensions for queries. In: CIKM (2011)
9. Feddoul, L., Schindler, S., Löffler, F.: Automatic facet generation and selection over knowledge graphs. In: Acosta, M., Cudré-Mauroux, P., Maleshkova, M., Pellegrini, T., Sack, H., Sure-Vetter, Y. (eds.) SEMANTiCS 2019. LNCS, vol. 11702, pp. 310–325. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-33220-4_23
10. Ferré, S.: Expressive and scalable query-based faceted search over SPARQL endpoints. In: Mika, P., et al. (eds.) ISWC 2014. LNCS, vol. 8797, pp. 438–453. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11915-1_28
11. Gipp, B., et al.: Scienstein: a research paper recommender system. In: ICETCCT (2009)
12. Gordon, J., et al.: Modeling concept dependencies in a scientific corpus. In: ACL (2016)
13. Gordon, J., et al.: Structured generation of technical reading lists. In: BEA@NAACL (2017)
14. Hahn, R., et al.: Faceted wikipedia search. In: Abramowicz, W., Tolksdorf, R. (eds.) BIS 2010. LNBIP, vol. 47, pp. 1–11. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12814-1_1
15. Harth, A.: VisiNav: a system for visual search and navigation on web data. J. Web Sem. **8**, 348–354 (2010)
16. Heim, P., Ziegler, J., Lohmann, S.: gFacet: a browser for the web of data. In: IMC-SSW@SAMT, vol. 417 (2008)
17. Jardine, J.G.: Automatically generating reading lists. Ph.D. thesis, University of Cambridge, UK (2014)
18. Jiang, Z., Dou, Z., Wen, J.: Generating query facets using knowledge bases. IEEE Trans. Knowl. Data Eng. **29**(2), 315–329 (2017)
19. Kong, W., Allan, J.: Extracting query facets from search results. In: SIGIR (2013)
20. Koutrika, G., Liu, L., Simske, S.: Generating reading orders over document collections. In: ICDE (2015)
21. Li, I., et al.: What should i learn first: introducing lecturebank for NLP education and prerequisite chain learning. In: AAAI (2019)
22. Liang, C., et al.: Measuring prerequisite relations among concepts. In: EMNLP (2015)
23. Liang, C., et al.: Recovering concept prerequisite relations from university course dependencies. In: AAAI (2017)
24. Liang, C., et al.: Investigating active learning for concept prerequisite learning. In: EAAI (2018)
25. Lu, W., et al.: Concept extraction and prerequisite relation learning from educational data. In: AAAI (2019)

26. Mäkelä, E., Hyvönen, E., Saarela, S.: Ontogator—a semantic view-based search engine service for web applications. In: Cruz, I., et al. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 847–860. Springer, Heidelberg (2006). https://doi.org/10.1007/11926078_61
27. Oren, E., Delbru, R., Decker, S.: Extending faceted navigation for RDF data. In: Cruz, I., et al. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 559–572. Springer, Heidelberg (2006). https://doi.org/10.1007/11926078_40
28. Pan, L., et al.: Prerequisite relation learning for concepts in MOOCs. In: ACL (2017)
29. Ponte, J.M., Croft, W.B.: A language modeling approach to information retrieval. In: SIGIR (1998)
30. Roy, S., et al.: Inferring concept prerequisite relations from online educational resources. In: AAAI (2019)
31. Sayyadiharikandeh, M., et al.: Finding prerequisite relations using the wikipedia clickstream. In: WWW Companion (2019)
32. Schraefel, M.C., et al.: The evolving mspace platform: leveraging the semantic web on the trail of the memex. In: Hypertext (2005)
33. Shahaf, D., Guestrin, C., Horvitz, E.: Metro maps of science. In: KDD (2012)
34. Sun, Y., Han, J.: Mining Heterogeneous Information Networks: Principles and Methodologies. Morgan & Claypool (2012)
35. Talukdar, P. P. and Cohen, W.: Crowdsourced comprehension: predicting prerequisite structure in wikipedia. In: BEA@NAACL (2012)
36. Upadhyay, P., et al.: TeKnowbase: towards construction of knowledge-base of technical concepts. Technical report. http://arxiv.org/abs/1612.04988 (2016)
37. Upadhyay, P., et al.: Construction and applications of teknowbase: a knowledge base of computer science concepts. In: WWW Companion (2018)
38. Upadhyay, P., Bedathur, S., Chakraborty, T., Ramanath, M.: Aspect-based academic search using domain-specific KB. In: Jose, J.M., et al. (eds.) ECIR 2020. LNCS, vol. 12036, pp. 418–424. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45442-5_52
39. Wang, S. and Liu, L.: Prerequisite concept maps extraction for automatic assessment. In: WWW Companion (2016)
40. Wohlgenannt, G., et al.: Dynamic integration of multiple evidence sources for ontology learning. JIDM **3**(3), 243–254 (2012)
41. Yang, Y., et al.: Concept graph learning from educational data. In: WSDM (2015)