Disjunctive Sets of Phrase Queries for Diverse Query Suggestion

Ziyang Liao
Kyoto University
Kyoto, Japan
liao@dl.soc.i.kyoto-u.ac.jp

Keishi Tajima
Kyoto University
Kyoto, Japan
tajima@i.kyoto-u.ac.jp

ABSTRACT

This paper proposes a method of suggesting expanded queries that disambiguate the original Web query which has multiple interpretations. In order to produce a diverse set of queries including those corresponding to infrequent query intents, our method produces queries by extracting phrases connecting given query terms from a corpus. We use a corpus because infrequent query intents may not appear in query logs. We use phrase queries because we need sufficiently specific queries for retrieving pages corresponding to infrequent query intents out of many pages corresponding to popular query intents. Phrase queries usually have high accuracy but low recall. In order to also achieve high recall, we use a disjunction of many phrase queries as a query. Our method first produces many phrase queries by using term expansion and phrase extraction from a corpus, then group semantically similar phrases into clusters, and use each cluster as a disjunctive set of phrase queries.

KEYWORDS

Web search, query modification, query expansion, query refinement, query disambiguation, infrequent query intent

ACM Reference Format:
Ziyang Liao and Keishi Tajima. 2019. Disjunctive Sets of Phrase Queries for Diverse Query Suggestion. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI '19), October 14–17, 2019, Thessaloniki, Greece.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3350546.3352566

1 INTRODUCTION

Formulation of appropriate queries in Web search is often a difficult task. One of the factors of inappropriate queries is ambiguity. Ambiguous query that have multiple interpretations often result in page lists only including pages corresponding to interpretations not intended by the users. It typically happens when the query has both popular and infrequent interpretations, and the user intends the infrequent one. For example, suppose a user who wants to produce fertilizer from oranges issues a query "orange fertilizer". The result would be filled with pages about fertilizer for growing oranges.

To help users reformulating queries, most search engines provide query suggestions. They suggest several types of queries, but expanded queries, which add keywords to the original query, are especially useful for disambiguating the original query.

WI '19, October 14–17, 2019, Thessaloniki, Greece

ACM ISBN 978-1-4503-6934-3/19/10…$15.00
https://doi.org/10.1145/3350546.3352566

The popular search engines extract queries to suggest from query logs. However, query logs may not include queries appropriate for infrequent query intents. For example, when given a query "orange fertilizer", query logs may not include queries appropriate for users who want to produce fertilizer from oranges.

In this paper, we propose a method of suggesting a diverse set of queries including those corresponding to infrequent query intents. Instead of extracting queries from query logs, our method generates queries by expanding the original query with phrases connecting the query terms. We extract the phrases connecting query terms not from query logs but from the document corpus, such as a Web corpus. We expect that the corpus has more diverse information than query logs, and we can produce a more diverse set of queries.

For example, given a query "orange fertilizer", if we can find a phrase "orange can be used as fertilizer" in the corpus, we can suggest a phrase query "orange can be used as fertilizer". We often manually produce this kind of phrase queries in order to disambiguate the original query, but our method automates it. Phrase queries requiring several query terms to appear in a specific order are supported by most major search engines.

Such phrases are useful for disambiguating the queries because it clarifies what relationship the query terms have in the context intended by the user, and it also clarifies the intended meaning of each term by placing it in a specific context. Phrase queries are especially useful for infrequent query intents because we need sufficiently specific queries for retrieving Web pages corresponding to the infrequent query intent out of many Web pages corresponding to other more frequent intents.

Phrase queries are very specific and can have high accuracy but usually have low recall. Our goal is, however, to produce a diverse set of queries to improve the recall of query suggestion. In order to achieve high recall, we extract many phrases from the corpus. To further increase the number of phrases we extract, we expand each query term to a set of its synonyms, hypernyms, and hyponyms. For example, given a query consisting of two terms t_1^0 and t_2^0, we first expand them to sets of terms $\{t_1^0, \ldots, t_1^n\}$ and $\{t_2^0, \ldots, t_2^m\}$, and extract connecting phrases for all combinations of (t_1^i, t_2^j).

Term expansion is useful for finding phrases corresponding to infrequent query intents because such phrases appear only infrequently, and may appear only with some of the synonyms, hypernyms, or hyponyms. For example, if we expand query terms "orange fertilizer" to "fruit fertilizer" and "apple fertilizer", we have a better chance of finding phrases related to the infrequent intent such as "can be made into". We then use it to expand the original query "orange fertilizer" into "orange can be made into fertilizer".

We produce many phrase queries for high recall, but if we simply show all of them to users, it would be very time-consuming for the users to identify appropriate queries among them and try

them one by one. In order to avoid it, we cluster produced phrase queries based on their semantic similarity, and show exemplars representing the clusters to the users. The users then choose one of them, and our method retrieves pages by using a disjunction of all phrases in the chosen cluster. By using a disjunction of many phrase queries, we can achieve both high accuracy and high recall.

Most queries submitted to Web search engines include only one or two terms [9, 22]. In the query logs of AOL recorded over three months in 2006, 19.4% of queries are single term and 30.5% of queries have two terms [15]. In this paper, we focus on queries consisting of two query terms. Generalization of our method to queries including three or more terms is an interesting remaining research issue.

The remainder of this paper is organized as follows. Section 2 overviews some related work. Section 3 describes the proposed method in detail. Section 4 explains the experiment and evaluation of the proposed method. Section 5 concludes the paper.

2 RELATED WORK

There have been much research on how to produce queries for query suggestion. Existing methods can be classified into two types: those that extract queries from query logs, and the others. We first explain the former and then explain the latter. We also discuss some other related research at the end of this section.

2.1 Query Suggestion using Query Logs

Most web search engines record queries submitted by users. There have been many proposals of methods that use this information for query suggestion. Baeza-Yates et al. [1] proposed a method for finding queries related to the current query from query logs. Their method ranks them based on two criteria: similarity to the current query, and how much the queries attracted the users in the past. Their method thus suggests queries that were popular in the past.

Jones et al. [10] proposed a method that uses the query sequences by other users in the past recorded in query logs. Cao et al. [4] also proposed a method that uses similar information. They first extract "concepts", which corresponds to sets of similar queries, from a query log by clustering a query-URL bipartite graph. When a user submits a sequence of queries, it is matched with the sequence of concepts in the query log to suggest the next query in the sequence.

There have also been proposal of methods for finding similar queries from query logs. Li et al. [13] proposed a method that finds queries that have co-occurred in a single session with the current query. Cucerzan and White [5] proposed a method that finds similar queries based on user landing pages, i.e., pages where users finally stopped their navigation started after the query. Ma et al. [15] also proposed a method for computing similarity between queries by using two bipartite graphs extracted from the click-through data: a user-query graph and a query-URL graph.

2.2 Query Suggestions without Query Logs

There have been a few studies on query suggestion methods that do not require query logs. Feuer et al. [7] proposed a method that extracts terms to add to the given query from the corpus. Their method extracts a term that most frequently appears in the close proximity of the query terms in the original query. Bhatia et al. [3] also proposed a method that extracts a term from the corpus. They rank the candidate terms based on the probability that the user would type that term following the given incomplete query. Shaikh et al. [19] also proposed a method that extracts terms based on the statistics obtained from Wikipedia. These methods suggest frequent terms and cannot diversify suggestions so that it also includes suggestions corresponding to infrequent query intents.

There have also been many studies on interactive auto-completion for search queries [2, 6, 20, 23]. Bast and Weber [2] proposed a method that extracts candidate terms from the search results of an incomplete query in real-time when a user types a query. Their method also suggests only the most frequent terms.

Kim et al. [12] proposed a method of query suggestion for academic paper search. Their method extracts subject-specific phrases used for describing ideas in academic papers. On the other hand, our method extracts phrases representing the relationship between two query terms in general documents.

2.3 Other Related Research

There have also been a few studies on how to show query suggestions in Web search. Shokouhi et al. [21] proposed a method that injects results retrieved by suggested queries directly into the result of the original query. On the other hand, our method groups suggested queries into clusters, and retrieves pages by using a disjunction of all queries in the cluster chosen by the user.

Kim et al. [11] proposed a method for diversifying query suggestions. Their target is tasks where the initial query is a whole document. Their method suggests queries covering as many aspects in the document. On the other hand, our goal is to suggest diverse queries including those corresponding to infrequent query intents.

Liu et al. [14] has shown that query suggestion is more useful for difficult queries, and we should target mainly on difficult queries. We mainly target on queries corresponding to infrequent query intents, which are one of the important classes of difficult queries.

3 PROPOSED METHOD

As explained before, the overview of our method is as follows:

(1) Given a query consisting of two terms t_1^0 and t_2^0, we expand them into sets of related terms $\{t_1^0, \ldots, t_1^n\}$ and $\{t_2^0, \ldots, t_2^m\}$,
(2) we extract connecting phrases $p_1, \ldots, p_l$ from the corpus by using all the combinations of (t_1^i, t_2^j),
(3) we generate phrase queries by connecting the original term t_1^0 and t_2^0 with the extracted phrases $p_1, \ldots, p_l$,
(4) we cluster generated expanded queries, choose an exemplar for each cluster, and show them to the user,
(5) the user choose an exemplar, and the system retrieves Web pages by using a disjunction of all the phrase queries in the cluster corresponding to the chosen exemplar.

In this section, we explain the details of each of these steps.

3.1 Query Term Expansion

When given a query "orange fertilizer", if we only use these original terms to find the phrases related to its infrequent query intents from a corpus, we may not obtain enough results since pages corresponding to the infrequent query intents in the corpus may not

use the combination of these words. We have this problem especially when some of the query terms are infrequent terms. Prior to extracting phrases from the corpus, we expand the query terms to increase the number of terms used for phrase extraction, and also to include more common terms in them. In the current implementation of our method, we use a web API known as WordsAPI[1] to obtain terms that are in the relation of synonyms, hasTypes, typeOf, hasInstances, and instanceOf with the given term. In the WordsAPI these relation have the following definition:

(1) synonyms: Words that can be interchanged for the original word in the same context.
(2) typeOf: Words that are more generic than the original word. Also known as hypernyms.
(3) hasTypes: Words that are more specific than the original word. Also known as hyponyms.
(4) instanceOf: Words that the original word is an example of.
(5) hasInstances: Words that are examples of the original word.

This API mainly uses the data found on WordNet [16] in order to obtain terms that have these relationship with the given term. When given a two-word query (t_1^0, t_2^0), we obtain sets of related words $\{t_1^0, \ldots, t_1^n\}$ and $\{t_2^0, \ldots, t_2^m\}$ by using this API.

3.2 Relation extraction

After expanding the query terms to $\{t_1^0, \ldots, t_1^n\}$ and $\{t_2^0, \ldots, t_2^m\}$, we use all the combinations of (t_1^i, t_2^j) to extract from the corpus the phrases connecting them, in other words, phrases representing the relationship between them. To extract them, we use Open Information Extraction (Open IE) [18] and ClueWeb12 corpus. Given "orange" and "fertilizer", Open IE returns phrases connecting them in this order, such as "require" and "can be made into", and also their frequency. Open IE does not simply extract word sequences occurring between two terms, but it parses the sentences and extract semantically meaningful phrases. Given "orange" and "fertilizer", we also extract phrases connecting them in the opposite order, such as "can grow" in "fertilizer can grow orange".

We then produce phrase queries by connecting obtained phrases to the original query terms instead of the expanded query terms. Let the result of the query to OpenIE be $P = \{(p_1, f_1), \ldots, (p_l, f_l)\}$, where p_i are the extracted phrases, and f_i are the occurrence frequency of p_i. We then create phrase queries "$t_1^0 p_1 t_2^0$", …, "$t_1^0 p_l t_2^0$". We also use $f_1, \ldots, f_l$ later for ranking clusters.

3.3 Phrase clustering method

We produce as many query suggestions as possible, but it is time-consuming for the user to examine all of them. In addition, there must be many queries without any matching Web pages. To help the user examine and use the suggested queries, we remove queries without matching pages in advance, and also cluster the remaining queries based on their semantic similarity.

We first use Siamese recurrent architectures by Mueller et al. [17] for learning semantic sentence similarity between phrases. It presents a Siamese adaptation of the Long Short-Term Memory (LSTMs) network for labeled data comprised of pairs of variable-length sequences and provides word-embedding vectors which are supplemented with synonymic information to the LSTMs that uses a fixed size vector to encode the underlining meaning expressed in a sentence (irrespective of the particular wording/syntax).

We then use an implementation of the Affinity Propagation Algorithm in scikit-learn[2] for clustering. Affinity Propagation Algorithm creates clusters by sending messages between pairs of items until a convergence is reached. The message represents the suitability of one item to be the exemplar of the other, which is in turn updated in response to the values from other pairs. The updating happens until convergence at which point the final exemplars are chosen and thereafter the final clustering is obtained. The query set is then described using a small number of exemplars which are identified as most representative of other items in the same cluster. We use the exemplar of each cluster to represent the cluster.

An advantage of Affinity Propagation Algorithm is that it produces good exemplars of clusters, which will be used as the representatives of the clusters. In addition, Affinity Propagation corresponds to similarity which is not symmetric and does not satisfy triangle inequality, unlike ordinary clustering methods such as k-means method [8]. Another merits of Affinity Propagation method is that the clustering result does not depend on the initial state.

We cluster $\{p_1, \ldots, p_l\}$ by using Affinity Propagation Algorithm. We create clusters so that each cluster includes phrases extracted with the same type of term expansion. Let the result of the clustering be $C = \{c_1, c_2, ..., c_n\}$. Each cluster c_i contains phrases that have high semantic similarity, and have been extracted with the same type of term expansion.

Table 1: Weights for Scoring Expansion Types

expansion type e	weight value $w(e)$
self, hasTypes, hasInstances	3
synonym	2
typeOf, instanceOf	1

3.4 Ranking algorithm

After clustering the queries, we rank the clusters and show the top ten clusters to the user. We rank the clusters based on the score of a cluster c_i, denoted by $S(c_i)$, which is defined as below:

$$S(c_i) = w(r_i^1)w(r_i^2) \sum_{p_j \in c_i} f_j$$

where r_i^1 is the semantic relationship used for expanding the first query term when we extract phrases in c_i, r_i^2 is the semantic relationship used for expanding the second query term when we extract phrases in c_i, $w(r)$ is the weight of the semantic relationship r, and f_j is the frequency of phrases explained in Section 3.2. The values of $w(r)$ are shown in Table 1. Notice that each c_i consists of phrases extracted through the same type of term expansion, so r_i^1 and r_i^2 are uniquely determined for each c_i.

The function $w(r)$ is used to give different weights to term expansion with different types of relationship. In general, compared with synonymy and hyponym relations, the expansion with hypernym relations makes the range of meanings of term sets wider.

[1] https://www.wordsapi.com

[2] http://scikit-learn.org/stable/modules/clustering.html

Table 2: Queries for Experiment

id	query	common search intent	minor search intent
1-1	apple fertilizer	Information on fertilizer for apples	How to make apples into fertilizer
1-2	banana fertilizer	Information on fertilizer for bananas	How to make bananas into fertilizer
1-3	orange fertilizer	Information on fertilizer for oranges	How to make oranges into fertilizer
2-1	Kyoto bank	Information on the Kyoto Bank	Information on banks in Kyoto
2-2	Japan bank	Information on the Bank of Japan	Information on banks in Japan
2-3	China bank	Information on the China Bank	Information on banks in China
3-1	steak sauce	Information on sauce for steak	How to make steak into sauce
3-2	beef sauce	Information on sauce for beef	How to make beef into sauce
3-3	chicken sauce	Information on sauce for chicken	How to make chicken into sauce

Table 3: Terms That Have Semantic Relationships with "orange"

synonyms	typeOf	hasTypes	instances	instanceOf
orangish	spectral color	reddish orange	citrus bergamia	river
orangeness	chromatic color	bigarade	bergamot	
orange tree	chromatic colour	bitter orange	citrus nobilis	
orange river	spectral colour	temple orange tree	citrus sinensis	
	citrus	bitter orange tree	sour orange	
	citrus tree	temple orange	bergamot orange	
	pigment	tangor	marmalade orange	
	citrus fruit	sweet orange tree	seville orange	
	citrous fruit	sweet orange	king orangee	
	citrus	citrus aurantium		

Table 4: Clusters of Phrases for "orange fertilizer"

exemplar of cluster	phrases in cluster
is	is
	found in
	has
provides	contains
	offers
	provides
is runoff from	is runoff from
	are contaminated with
	finds a new home on
contain amounts of	contain amounts of
was a 50-50 mix of	was a 50-50 mix of

Therefore, we set the weight of the hypernym to be the lowest, the synonymous to be the second, and the weight of the hyponym to be the highest, which is the same weight as the original term.

4 EXPERIMENT

In this section, we explain the experiment we conducted for evaluating our method. We collected queries that have multiple interpretations, and compared query suggestions by our method for those queries with those by Google search engine and Bing search engine. We evaluated them based on whether the suggestion list includes at least one query whose top 10 search results include at least one Web page relevant to the given infrequent search intent.

4.1 Test Queries

The queries used in the experiment are listed in Table 2. These queries have multiple possible interpretations. For example, the most common query intent of the query "beef sauce" must be "information on sauce for beef", but its intent can also be "information on how to make beef into sauce". Similarly to the case of "orange fertilizer", the search result of the query "beef sauce" is filled with the pages about the popular intent "sauce for beef", and the users with the latter infrequent query intent need to reformulate their query to obtain the information they need.

4.2 Suggestion Query Generation

We first expand the query terms by using WordsAPI. For example, Table 3 shows the terms we obtained for the term "orange".

After expanding the query terms, we use the obtained term sets to extract from the web corpus the phrases representing the relation between two query terms. For example, Table 4 shows the result of the clustering of obtained phrases for the query "orange fertilizer".

After clustering the phrases, we rank the clusters, and we show the exemplars of the top ten clusters to the users as suggested queries. Table 5 shows the results. Table 5 also shows the query suggestion results of Google search engine.

4.3 Evaluation

We compare several variations of our method with different types of term expansion, including a method without term expansion. The compared methods are listed below:

Table 5: Query Suggestion Results of Google and Our Method

Query = apple fertilizer		Query = orange fertilizer	
Google	Our approach	Google	Our approach
apple tree fertilizer 10-10-10	apple ranked fertilizer	orange tree fertilizer florida	orange is fertilizer
what is the fertilization rate for an apple orchard	apple contains fertilizer	homemade fertilizer for citrus trees	fertilizer produced orange
fertilizing pear trees	apple wont grow fertilizer	citrus fertilizer	fertilizer transported in orange
when to fertilize plum trees	fertilizer are used on apple	how often to water orange trees	fertilizer end up in orange
apple tree fertilizer spikes	apple has about fertilizer	when to fertilize citrus trees in southern california	orange contains fertilizer
what should i feed my apple tree?	fertilizer target apple	citrus fertilizer npk	fertilizer arrow orange
when to fertilize peach trees		liquid citrus fertilizer with micronutrientsk	fertilizer pollute orange
apple tree fertilizer for sale		citrus fertilizer home depot	fertilizer found in orange
			orange has fertilizer
			orange made from fertilizer

Query = banana fertilizer		Query = Kyoto bank	
Google	Our approach	Google	Our approach
banana fertilizer npk	banana are fertilizer	bank of kyoto english	Kyoto is located in bank
banana fertilizer mix	fertilizer made with banana	bank of kyoto investor relations	Kyoto rests on bank
banana fertilizer schedule	banana contains fertilizer	bank of kyoto linkedin	Kyoto is home to bank
banana fertilizer management	banana purchase fertilizer	bank of kyoto address	Kyoto is built on bank
banana fertilizer home depot	fertilizer dried banana	bank of kyoto share price	Kyoto fargo bank
fertilizer for banana trees care	fertilizer is the only right way to grow banana	kyoto bank exchange rate	bank is Kyoto
homemade fertilizer for banana trees	fertilizer brought about by banana	bank of nagoya	Kyoto stretches along bank
banana fertilizer lowes	fertilizer are removed from banana	best money exchange kyoto	bank located in Kyoto
			bank is on Kyoto
			bank is one of Kyoto

Query = Japan bank		Query = China bank	
Google	Our approach	Google	Our approach
japan post bank	Japan is on bank	china bank branches	China is on bank
japan post bank internet banking	Japan has been following bank	china bank savings branches	China have bank
japan post bank atm	Japan kindly thanks bank	china bank savings online	China has raised bank
japan post bank account number	bank is acceptable only from Japan	china bank savings account	China is home to bank
japan post bank remittance	Japan does bank	china bank open account	China lowered bank
japan post bank branch code	bank said Japan	china bank banking hours	China reduced bank
japan post bank locations	bank would come from Japan	china bank branches open on a saturday	China has bank
japan post bank login	Japan entered the war on bank	china bank near me	China are bank
	Japan are bank		bank side in China
	Japan is located on bank		China wants to build bank

Query = steak sauce		Query = beef sauce	
Google	Our approach	Google	Our approach
creamy steak sauce	sauce is on steak	beef sauce for rice	beef beethickened in sauce
best steak sauce recipe	steak are served with sauce	creamy steak sauce	beef served with sauce
garlic steak sauce	steak make sauce	best steak sauce recipe	beef is served in sauce
balsamic steak sauce	sauce remove steak	garlic steak sauce	beef is sauce
mushroom steak sauce recipe	sauce is great for steak	beef recipes	sauce is served with beef
steak sauce brands	steak pour on sauce	mushroom steak sauce recipe	beef cooked in sauce
steak sauce without worcestershire	sauce has steak	sauces for beef roast	sauce cook beef
red wine steak sauce recipe	sauce pour over steak	diane sauce for steak	sauce had beef
	steak include sauce		beef marinated with sauce
	steak is made with sauce		beef add in sauce

Query = chicken sauce	
Google	Our approach
chicken sauces and marinades	chicken served in sauce
sauce for chicken and rice	sauce pour the mixture over chicken
how to prepare chicken sauce	chicken stir into sauce
creamy sauces for chicken	sauce combine chicken
tomato sauce for chicken	chicken are marinated in sauce
healthy sauces for chicken	chicken make sauce
butter sauce for chicken	chicken tossed in sauce
roast chicken sauce	chicken flavored sauce
	chicken combine sauce
	sauce be used to chicken

Table 6: Discovery of Queries Corresponding to Given Infrequent Search Intents

id	1-1	1-2	1-3	2-1	2-2	2-3	3-1	3-2	3-3
Google		✓			✓				
Bing		✓		✓	✓			✓	
1) without expansion					✓	✓	✓		✓
2) synonym expansion									
3) typeOf expansion		✓	✓						
4) instanceOf expansion									
5) hasTypes expansion									
6) hasInstances expansion									
7) full expansion without expansion type scoring	✓	✓	✓		✓	✓	✓	✓	✓
8) full expansion with expansion type scoring	✓	✓	✓		✓	✓	✓	✓	✓

(1) a method without term expansion,
(2) a method with synonym expansion,
(3) a method with typeOf expansion,
(4) a method with instanceOf expansion,
(5) a method with hasTypes expansion,
(6) a method with hasInstances expansion,
(7) a method with full (itself, synonym, typeOf, instanceOf, hasTypes, hasInstances) expansion and without the expansion type scoring, and
(8) a method with full (itself, synonym, typeOf, instanceOf, hasTypes, hasInstances) expansion and expansion type scoring.

The method with full expansion and without the expansion scoring (7) uses all types of expansions, but do not use the scoring function $S(c_i)$ and rank clusters simply by $\sum f_i$. On the other hand, the method with full expansion and with the expansion scoring (8) uses the cluster ranking by $S(c_i)$.

Table 6 shows the result of the evaluation. The method with full expansion and with the expansion type scoring (8) could suggest at least one query that could retrieve pages matching with the given infrequent query intent for 8 queries (all but the query 2-1), and the method with full expansion and without the expansion type scoring (7) also could for 8 queries. On the other hand, Google succeeded only for 2 queries and Bing succeeded only for 4 queries.

For these 9 test queries, no method with a single type of expansion is better than the method without expansion. On the other hand, the two methods with all expansion types are better than the others. It means that we need to combine multiple expansion types to extract useful phrases in these cases.

These results prove that:

- our approach which extracts phrases from Web corpus is useful for the query suggestion for infrequent search intents,
- query term expansion by using several types of semantic relationship improves the performance of our method, and
- our scoring function based on the type of expansion did not further improve nor degrade our method significantly.

5 CONCLUSION AND FUTURE WORK

In this paper, we proposed a method for producing diverse query suggestions including even those corresponding to infrequent query intents. Our method extracts phrases connecting given two query terms from the corpus, and produce query suggestions by connecting those phrases with the original query terms. In order to improve the recall of query suggestions, our method expand the original query terms to the set of related terms, and extract the connecting phrases by using these terms. By using all combinations of them, we can produce many suggestion queries, but we cluster them and produce a small number of disjunctive queries. By this strategy, we avoid the cost of users for examining all query suggestions one by one, and also achieves both high precision and high recall.

The experimental results show that our approach outperforms the query suggestion by Google and Bing in terms of the ability to include query suggestions corresponding to infrequent query intents. However, there are several aspects that have been out of the scope of this paper, which we discuss below.

(1) Our approach is limited to suggestion for original queries consisting of more than two terms. We need to generalize our method so that we can handle other kinds of queries.
(2) Our system sometimes generates phrase queries that are badly formed and hence are not meaningful as search queries (although they sometimes happen to yield appropriate results when used as search queries). One possible future goal would be to ensure that the badly formed combination of phrases are eliminated from the suggestions.
(3) Our approach is assuming queries that are difficult because of the ambiguity of the intended information need, especially queries where the relationship between two query terms is unclear. Usefulness of our method for other types of difficult queries should also be discussed and experimentally evaluated.
(4) In this research, we have primarily focused on the effectiveness of the queries suggested by our method, and did not focus on the efficiency of our algorithm. The computation cost of our method is not critical, but it will be interesting to explore how we can produce similarly effective query suggestions more efficiently than we currently do so that our method can be applied to a larger scale systems with huge number of queries as well.

6 ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI JP18H03245 and JST CREST JPMJCR16E3, Japan.

REFERENCES

[1] Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. 2004. Query Recommendation Using Query Logs in Search Engines. In *Proceedings of the 2004 International Conference on Current Trends in Database Technology*. Springer-Verlag, 588–596. https://doi.org/10.1007/978-3-540-30192-9_58

[2] Holger Bast and Ingmar Weber. 2007. The CompleteSearch engine: Interactive, efficient, and towards IR & DB integration. In *Third Biennial Conference on Innovative Data Systems*. 88–95.

[3] Sumit Bhatia, Debapriyo Majumdar, and Prasenjit Mitra. 2011. Query suggestions in the absence of query logs. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 795–804.

[4] Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. 2008. Context-aware query suggestion by mining click-through and session data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 875–883.

[5] Silviu Cucerzan and Ryen W White. 2007. Query suggestion based on user landing pages. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 875–876.

[6] Giovanni Di Santo, Richard McCreadie, Craig Macdonald, and Iadh Ounis. 2015. Comparing Approaches for Query Autocompletion. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 775–778.

[7] Alan Feuer, Stefan Savev, and Javed A Aslam. 2007. Evaluation of phrasal query suggestions. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. ACM, 841–848.

[8] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. 1999. Data clustering: a review. *ACM computing surveys (CSUR)* 31, 3 (1999), 264–323.

[9] Bernard J. Jansen, Amanda Spink, Judy Bateman, and Tefko Saracevic. 1998. Real Life Information Retrieval: A Study of User Queries on the Web. *SIGIR Forum* 32, 1 (1998), 5–17. https://doi.org/10.1145/281250.281253

[10] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*. ACM, 387–396.

[11] Youngho Kim and W. Bruce Croft. 2014. Diversifying Query Suggestions Based on Query Documents. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*. 891–894.

[12] Youngho Kim, Jangwon Seo, W. Bruce Croft, and David A. Smith. 2014. Automatic suggestion of phrasal-concept queries for literature search. *Information Processing & Management* 50, 4 (2014), 568–583.

[13] Yanan Li, Bin Wang, Sheng Xu, Peng Li, and Jintao Li. 2009. Querytrans: Finding similar queries based on query trace graph. In *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT'09. IEEE/WIC/ACM International Joint Conferences on*, Vol. 1. IEEE, 260–263.

[14] Yang Liu, Ruihua Song, Yu Chen, Jian-Yun Nie, and Ji-Rong Wen. 2012. Adaptive Query Suggestion for Difficult Queries. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 15–24.

[15] Hao Ma, Haixuan Yang, Irwin King, and Michael R Lyu. 2008. Learning latent semantic relations from clickthrough data for query suggestion. In *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 709–718.

[16] George A Miller. 1995. WordNet: a lexical database for English. *Commun. ACM* 38, 11 (1995), 39–41.

[17] Jonas Mueller and Aditya Thyagarajan. 2016. Siamese Recurrent Architectures for Learning Sentence Similarity. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. 2786–2792. http://dl.acm.org/citation.cfm?id=3016100.3016291

[18] Swarnadeep Saha, Harinder Pal, and Mausam. 2017. Bootstrapping for Numerical Open IE. In *ACL*.

[19] M. T. Shaikh, M. S. Pera, and Y. K. Ng. 2013. A Probabilistic Query Suggestion Approach without Using Query Logs. In *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*. 633–639.

[20] Milad Shokouhi. 2013. Learning to Personalize Query Auto-completion. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 103–112.

[21] Milad Shokouhi, Marc Sloan, Paul N. Bennett, Kevyn Collins-Thompson, and Siranush Sarkizova. 2015. Query Suggestion and Data Fusion in Contextual Disambiguation. In *Proceedings of the 24th International Conference on World Wide Web*. 971–980.

[22] Craig Silverstein, Hannes Marais, Monika Henzinger, and Michael Moricz. 1999. Analysis of a Very Large Web Search Engine Query Log. *SIGIR Forum* 33, 1 (Sept. 1999), 6–12. https://doi.org/10.1145/331403.331405

[23] Stewart Whiting and Joemon M. Jose. 2014. Recent and Robust Query Auto-completion. In *Proceedings of the 23rd International Conference on World Wide Web (WWW '14)*. 971–982.