

A COMMUNICATION SERVER, A METHOD, A USER DEVICE, AND A SYSTEM

Technical Field

The invention relates generally to the field of communication. One aspect of the invention relates to a communication server apparatus for ranking search results. Another aspect of the invention relates to a method, performed in a communication server apparatus for ranking search results. Another aspect of the invention relates to a communication device for ranking search results. Another aspect of the invention relates to a communication system.

Background

Various forms of ranking search results exist.

For example, US8898583 describes a system to show some highlighted semantic entities on a webpage that a user is browsing. The user can then indicate their desire on any entity (e.g., by hovering a mouse on that) to trigger a contextual search of that entity. US8010523 also discloses methods of ranking search results. One technical problem that may exist in the art is how to rank the relevant results based on the user's behavioural preferences, compared to other users who share similar intentions.

Summary

Embodiments may be implemented as set out in the independent claims. Some optional features are defined in the dependent claims.

Implementation of the techniques disclosed herein may provide significant technical advantages. Advantages of one or more aspects may include:

- more accurate ranking of results for each user;
- more accurate correlation between user's intention and ranking results;
- more accurate correlation between user's behaviour and ranking results;
- higher click-through rate: users will click more when the search results shown to them are more relevant;

higher conversion rate: conversions (i.e., orders) will increase when users find more relevant candidates;

better user experience: user will be more satisfied with the searching results and hence the whole service; and/or

more engagement: when users are more satisfied, they are likely to engage more (by browsing more on the app, or using the app more frequently).

In at least some implementations, the techniques disclosed herein may allow for:

- the technical solution of reduced greenhouse emissions based on the technical problem of ranking search results. Because the ranking is more accurate less searching needs to be done and therefore less greenhouse gases are emitted in generating power for powering the servers and/or powering the server cooling system;
- the technical solution of reduced greenhouse emissions based on the technical problem of ranking search results. Because the ranking is more accurate there may be less wasted trips by drivers, or less wasted sales by merchants, avoiding the need for a second sale to a better merchant and therefore greenhouse emissions for any unnecessary trips or unnecessary product manufacturing may be avoided;
- the technical solution of reduced data centre energy requirements based on the technical problem of ranking search results. Because the ranking is more accurate less searching needs to be done and therefore less energy is required for powering the servers and/or for cooling the servers;
- the technical solution of reduced server hardware required for the technical problem of ranking search results. Because the ranking is more accurate less searching needs to be done;
- the technical solution of reduced bandwidth requirements based on the technical problem of ranking search results. Because the ranking is more accurate less searching needs to be done and therefore less bandwidth is required for communications;
- the technical solution of faster execution of search results for the technical problem of how to rank search results;

- the technical solution of better ranking of search results based on the technical problem of how to achieve a higher click-through rate. Users are likely to click more because the ranking results are more relevant and accurate;
- the technical solution of better ranking of search results based on the technical problem of how to achieve a higher conversion rate. Because the ranking results are more relevant and accurate, it will eventually result in a higher number of conversions;
- the technical solution of better ranking of search results based on the technical problem of how to achieve a better user experience. Users will be more satisfied with the searching results and hence the whole service provided to them; and/or
- the technical solution of better ranking of search results based on the technical problem of how to achieve more user engagement. When users find the ranking results are more relevant and satisfying, they are likely to engage more by browsing more contents, placing more orders and using the application more frequently.

In an exemplary implementation, the functionality of the techniques disclosed herein may be implemented in software running on a server communication apparatus (such as a cluster of servers or a cloud computing platform), which communicates with the applications running on the terminals, such as mobile phones. The software which implements the functionality of the techniques disclosed herein may be contained in a computer program, or computer program product. The server communication apparatus establishes secure communication channels with the user terminals for receiving the queries from users and rendering the search ranking results to the users. The process also includes the processing of queries, fetching data from databases and execution of ranking strategies in order to present a ranked list of results to the users.

According to a further aspect there is provided a communications server apparatus for ranking search results, the communications server comprising a processor and a memory, the communications server apparatus being configured, under control of the processor, to execute instructions stored in the memory, to:

- estimate and store a plurality of search intent identifiers at one or multiple levels;
- estimate and store behavioural signals corresponding to the search intent identifiers;

receive a user search query;
computationally map the user search query to the search intent identifiers;
determine and fetch the corresponding behavioural signals from the storage, based on the mapped search intent identifiers; and
rank a set of search results based on the user search query, the mapped search intent identifiers and the corresponding behavioural signals.

A first level of the multiple levels of search intent identifiers may be one or more keyword identifiers.

The server apparatus may include a processing unit configured to convert the user search query to keyword IDs.

The server apparatus may be further configured to update the plurality of keyword identifiers and the corresponding behavioural signals stored in the database based on the keywords having frequencies above some predetermined thresholds, used in all users' search queries over a predetermined time period.

A second level of the multiple levels of search intent identifiers may be one or more estimated entity identifiers.

The server apparatus may include an entity recognition unit configured to extract the estimated entity identifiers from the user search query.

The server apparatus may include an entity fusion/pooling unit configured to concatenate/extract entity identifiers from multiple recognised entities.

A third level of the multiple levels of search intent identifiers may be one or more estimated topic identifiers.

The server apparatus may be further configured to parse the user search query into a series of tokens.

The server apparatus may be further configured to generate an embedding of the series of tokens.

The server apparatus may be further configured to cluster and map the embedding to the estimated topic identifiers.

The behavioural signals may be derived from the user behavioural events in relation to previous search results.

The user's behavioural events may be selected from the group consisting of view behaviours, click behaviours, order behaviours, dwell time, any behaviours that related to the search, and any combination thereof.

The behavioural signals may be obtained by mathematical computations/aggregations of the user behavioural events based on one kind of search intent identifiers or any combination between some kinds of search intent identifiers and other identifiers.

The behavioural signals may be obtained by mathematical computations/aggregations of the user behavioural events, which are performed through regular manner, real-time manner or combination of both.

The search intent identifiers and corresponding behavioural signals may be stored in an offline database and/or a database with low writing and reading latency which will be used for online model serving.

The ranking of the search results may also be based on user features, candidate features, any other features that relate to the search, and any combination thereof.

Brief Description of the Drawings

The invention will now be described, by way of example only, and with reference to the accompanying drawings in which:

Fig. 1 is a schematic block diagram illustrating an exemplary delivery/transportation service.

Fig. 2 is a schematic block diagram illustrating an exemplary communications server for the delivery/transportation service.

Fig. 3 is an example user interface (UI) for a search ranking of merchants.

Fig. 4 is a schematic diagram of an example search ranking system.

Fig. 5 is a schematic diagram of an example wide and deep model architecture.

Fig. 6 is a schematic diagram of an example model architecture for search ranking of merchants.

Fig. 7 is a schematic diagram of a system overview according to an example embodiment.

Fig. 8 is a schematic diagram of the Multi-level intention mapping in Figure 7.

Fig. 9 is a graph of the cumulative frequency of candidate keywords in searched queries.

Fig. 10 is a schematic diagram of an example framework to train an entity recognition model to convert keywords to embedding.

Fig. 11 is a schematic diagram of an example framework to train a topic recognition model.

Fig. 12 is a schematic diagram of user behavioural signal aggregation.

Fig. 13 is a database schema for a non-relational database to store behavioural signals for model serving.

Fig. 14 is a schematic diagram of model training using behavioural signals.

Detailed Description

The techniques described herein are described primarily with reference to use in ranking search results for applications including, but not limited to, food deliveries, online groceries, taxi, ride hailing, ride sharing, tickets, attractions, vouchers, service / trade exchanges, and pet transport. It will be appreciated that these techniques may have a broader reach and may be usefully implemented in other fields where ranking search results may be required. Generally, this might be the case in any data search request.

Figure 1 shows an exemplary architecture of a delivery/transportation service system 100, with a number of users each having a communications device 104, a number of merchants each having a communication device 109, a number of drivers each having a user interface communications device 106, a server 102 (or geographically distributed servers) and

communication links 108 connecting each of the components. Each user contacts the server 102 using a user software application (app) on the communications device 104.

For deliveries, the user app may allow the users to input queries containing the keywords for the items of interest and delivery addresses. The user may see a list of merchants and/or items provided by the merchants, and order items from the merchants. The merchants contact the server 102 using the merchant app for providing the information about their items and receiving orders for each confirmed transaction. The drivers contact the server 102 using the driver app on the communication device 106. The driver app allows the drivers to indicate their availability to take the delivery jobs, information about their vehicle, their location. The server 102 may then match the orders to drivers, based on, for example, geographic location of the drivers, merchant addresses and delivery addresses, driving conditions, traffic level / accidents, demands and supplies, etc.

For transportation, the user app may allow the user to enter their pick-up location, a destination address, one or more service parameters, and/or after-ride information such as a rating. The one or more service parameters may include the number of seats of the vehicle, the style of vehicle, level of environmental impact and/or what kind of transport service is desired. Each driver contacts the server 102 using a driver app on the communication device 106. The driver app allows the driver to indicate their availability to take the ride jobs, information about their vehicle, their location, and/or after-ride info such as a rating. The server 102 may then match users to drivers, based on, for example: geographic location of users and drivers, maximising revenue, user or driver feedback ratings, weather, driving conditions, traffic level / accidents, relative demand, environmental impact, and/or supply levels. This allows an efficient allocation of resources because the available fleet of drivers is optimised for the users' demand in each geographic zone.

Referring to Figure 2, further details of the components in the system of Figure 1 are now described. The communication apparatus 100 comprises the communication server 102, and it may include the user communication device 104, the merchant communication device 109 and the driver communication device 106. These devices are connected in the communication network 108 (for example, the Internet) through respective communication

links 110, 111, 112, 114 implementing, for example, internet communication protocols. The communication devices 104, 106 and 109 may be able to communicate through other communication networks, including mobile cellular communication networks, private data networks, fibre optic connections, laser communication, microwave communication, satellite communication, etc., but these are omitted from Figure 2 for the sake of clarity.

The communication server apparatus 102 may be a single server as illustrated schematically in Figure 2. Alternatively, the functionality performed by the server apparatus 102 may be distributed across multiple physically or logically separate server components. In the example shown in Figure 2, the communication server apparatus 102 may comprise a number of individual components including, but not limited to, one or more microprocessors 116, a memory 118 (e.g., a volatile memory such as a RAM, and/or longer term storage such as SSD (Solid State or Hard disk drives (HDD)) for the loading of executable instructions 120, the executable instructions defining the functionality the server apparatus 102 carries out under control of the microprocessor 116. The communication server apparatus 102 also comprises an input/output module 122 allowing the server to communicate over the communication network 108. User interface 124 is provided for user control and may comprise, for example, computing peripheral devices such as display monitors, computer keyboards and the like. The server apparatus 102 also comprises a database 126, the purpose of which will become readily apparent from the following discussion.

The user communication device 104 may comprise a number of individual components including, but not limited to, one or more microprocessors 128, a memory 130 (e.g., a volatile memory such as a RAM) for the loading of executable instructions 132, the executable instructions defining the functionality the user communication device 104 carries out under control of the microprocessor 128. The user communication device 104 also comprises an input/output module 134 allowing the user communication device 104 to communicate over the communication network 108. A user interface 136 is provided for user control. If the user communication device 104 is, say, a smartphone or tablet device, the user interface 136 will have a touch panel display as is prevalent in many smartphones and other handheld devices. Alternatively, if the user communication device 104 is, say, a

desktop or laptop computer, the user interface 136 may have, for example, computing peripheral devices such as display monitors, computer keyboards and the like.

The merchant communication device 109 may be, for example, a smartphone or tablet device with the same or a similar hardware architecture to that of the user communication device 104.

The driver communication device 106 may be, for example, a smartphone or tablet device with the same or a similar hardware architecture to that of the user communication device 104. Alternatively, the functionality may be integrated into a bespoke device such as a taxi fleet management terminal.

Thus, it will be appreciated that Figures 1 and 2 and the foregoing description illustrate and describe a communication server apparatus 102 comprising a microprocessor 116 and a memory 118, the communication server apparatus 102 being configured, under control of the microprocessor 116, to execute instructions 120 stored in the memory 118, to:

- store a plurality of sets of search intent identifiers, wherein respective sets of search intent identifiers are associated with different respective levels of semantic granularity, and wherein a first set of the search intent identifiers is a set of keyword IDs;

- receive a user search request;

- computationally map the user search request to one or more estimated keywords, the estimated keywords selected from the set of keyword IDs;

- computationally map the one or more estimated keywords to one or more estimated first derived features, the estimated first derived features selected from a second set of the search intent identifiers that is not the set of keyword IDs;

- determine one or more behavioural signals based on the one or more estimated keywords and/or the one or more estimated first derived features; and

- rank a set of search results based on the user search request and the one or more behavioural signals.

Further, it will be appreciated that Figures 7 and 8 illustrate and describe a method performed in a communication server apparatus 102, the method comprising, under control of a microprocessor 116 of the server apparatus 102:

- storing a plurality of sets of search intent identifiers, wherein respective sets of search intent identifiers are associated with different respective levels of semantic granularity, and wherein a first set of search intent identifiers is a set of keyword IDs;

- receiving a user search request;

- computationally mapping the user search request to one or more estimated keywords, the estimated keywords selected from the set of keyword IDs;

- computationally mapping the one or more estimated keywords to one or more estimated first derived features, the estimated first derived features selected from a second set of search intent identifiers that is not the set of keyword IDs;

- determining one or more behavioural signals based on the one or more estimated keywords and/or the one or more estimated first derived features; and

- ranking a set of search results based on the user search request, and the one or more behavioural signals.

Estimated keywords, estimated first derived features, and the later discussed estimated second derived features may be generalised as search intent identifiers, and may be chosen according to the requirements of the application. For example, in the application of food deliveries, the first derived feature may be chosen as an estimated entity selected from a set of entity IDs. Each entity ID may represent a different merchant stored within the system. The second derived feature may be chosen as an estimated topic selected from a set of Topic IDs. Each topic ID may represent one of a number of generalised topics, which user search queries commonly relate to. Other example of the search intent identifiers may include sector ID and domain ID, etc. For example, in the use case of a document search, a sector ID can represent a sector, e.g., "electronics", while a domain ID can represent a more general domain, e.g., "engineering".

Internet users constantly rely on search requests to find the information they want from the internet, from the database, private networks or other data repositories. For example, as shown in Figure 3 users may use the search functionality to find food merchants and dishes

of interest and perform checkouts from the ones they desire in a food delivery app. Accuracy of the search and ranking is therefore an important component to maximising efficiency of searching and may therefore drive significant traffic and revenue.

An example search engine is shown in Figure 4. It includes two stages: recall 402 and ranking 404. Given a search request containing keywords, the recall stage 402 firstly finds a number of relevant candidates using, for example, ElasticSearch. The top K recalled candidates are then passed to the ranking stage 404 where machine learning (ML) models and personalized features are involved to achieve higher precision for the ranking results.

For ranking stage 404, possible ML approaches include the click-through-rate (CTR) models, including GBDT+LR, Wide and Deep, DeepFM, and DIN. In addition, the learning-to-rank (L2R) models may also be used, and an exemplary implementation framework can be the Tensorflow-Ranking framework. In terms of the features, the user features (e.g., age, gender, purchasing level, average past order values, etc.), candidate features (e.g., category, regional popularity, etc.), as well as user-candidate crossing features (e.g., user-candidate orders, etc.), can be incorporated in the model.

For the search scenario, a user often comes with a strong intention when typing a query of keywords. Such keywords can often be short-term, like, “thai”, “din-tai-fung”, “chicken-rice”, etc. Although the recall stage will find the best matched candidates based on the keywords, whether the ranking stage can further exploit such keywords information to interpret the user's intention and rank the corresponding candidates to the top can be important for the final user conversion.

The L2R framework may be used as the main ranking framework, with the list-wise ranking strategy. That is, the ranking model is trained given a number of query retrieves, and for each retrieve, a list of candidates is ranked via a list-wise softmax function. The model architecture may be the Wide and Deep architecture (<https://arxiv.org/abs/1606.07792>) where features can be input to wide or deep sides, as shown in Figure 5 (where the wide side of the architecture is shown at left, and the deep side at right).

Mathematically, the model's prediction is given in equation 1:

$$P(Y = 1|\mathbf{x}) = \sigma(\mathbf{w}_{wide}^T[\mathbf{x}, \phi(\mathbf{x})] + \mathbf{w}_{deep}^T a^{(l_f)} + b) \quad (1)$$

where Y is the binary class label, x stands for the original features, $\sigma(.)$ is the sigmoid function and b is the bias term. On the wide side, $\phi(.)$ stands for the cross product transformation (e.g., "AND(is_promo=1, purchasing_level=2)") for the original features x , and w_{wide} are the trainable weights for the wide side. On the deep side, the features are passing through a few fully connected layers as given in equation 2:

$$a^{(l+1)} = f(W^{(l)}a^{(l)} + b^{(l)}) \quad (2)$$

and $a^{(l_f)}$ are the activations at the final layer, w_{deep} are the trainable weights for the last layer.

To exploit the keyword information, a two-tower neural network (NN) architecture can be used. The full model architecture 600 is shown in Figure 6. The searched keyword is firstly tokenized through a trained tokenizer 602. The tokenized keyword is then passed through a textCNN model 604 to get the keyword embedding (a learned dense vector which represents the underlying information). The candidate_id is passed through an embedding layer 606 and finally, the cosine similarity 608 is applied after a few fully-connected layers on both sides.

The architecture 600 in Figure 6 may be adapted to encompass multiple languages and multiple countries. Each additional language within each country would have its own additional model 610, although a language model could be shared between multiple countries if the intelligibility between that language in various countries was high enough. Each language model could be considered as using different corpus together in training a single holistic model, which is then used on live search queries to serve customers in multiple countries.

This method exploits the textual relevance between the keywords and the candidates. For example, it could find a high relevance between the keywords "vegan pizza" and the merchants "12anadian pizza" and "pizza hut", because both merchants mainly sell pizza. However, this model may not extract user behavioural preferences indicating that a

particular merchant “sunny slices” is actually the best choice, because it sells a particular kind of vegan pizza that is most people’s favourite in that city.

So understanding the user’s underlying intentions and behavioural patterns may be beneficial depending on the requirements of the application. A purely machine learning model without careful consideration of behavioural signals may not provide optimum results.

One embodiment proposes a framework to exploit the user behavioural signals in the search context through a multi-level topology. The basic topology is at the user-level where the user behavioural events are aggregated on user_id to obtain the behavioural signals pertaining to a specific user. In addition to this, the framework may include three hierarchical levels of intention mapping, as shown in Fig. 7 and 8.

The first one is called keyword-level user intention mapping, which is also the most granular level in the search context. The searched keywords are converted to keyword_id upon which the behavioural events (which can be among users) pertaining to the same keyword_id will be aggregated to obtain the behavioural signals at keyword level.

The second level is called entity-level user intention mapping where the searched keywords are mapped to entities (e.g., cuisine name, dish name, etc.). The behavioural events (which can be among users) pertaining to the same entity can then be aggregated upon that entity to obtain the behavioural signals at entity level. This implies the aggregation is on a more general basis than the keyword level.

The third level is called topic-level user intention mapping where the searched keywords are computationally mapped to topics. The behavioural events (which can be among users) pertaining to the same topic can then be aggregated upon that topic to obtain behavioural signals at topic level. This is the highest level of the behavioural events aggregation and the behavioural signals are obtained on the most general basis.

For the behavioural events aggregation, the proposed framework can execute the aggregation job not only in a regular manner (e.g., daily) but also in a real-time manner as shown in 720 and 716, respectively, in Fig. 7 and will be discussed in more detail later. As a result, the behavioural events can be aggregated at the aforementioned levels to form both the offline and real-time behavioural signals. Exploitation of real-time behavioural signals may improve the ranking results by capturing the short-time user preference.

According to various embodiments, the explicit exploitation of such behavioural signals to capture the underlying behavioural patterns could improve the ranking of search results compared to the use of a pure ML model that exploits only the textual relevance as introduced before.

According to some embodiments, the proposed technique exploits a granular-to-general user intention mapping methodology to convert the user's intention to different keys (i.e., keyword_id, entity_id, and topic_id) so that the user preference can be quantified as signals at different levels. According to some embodiments, there could be some advantages in this methodology. At a granular level, a common behavioural pattern among users helps the prediction task. Take keyword-level for example, the popular restaurant choices among users who searched for "thai-food" can be utilized to help the ranking task when the same exact keyword query comes next time. In addition, ranking can be further improved by considering behavioural patterns at higher levels. For example, the different queries "thai-food", "thai" and "thai-restaurant" can correspond to the same entity as "cuisine:thai" so that the popular restaurant choices among users who searched for the same entity can be utilized to help the ranking task when these similar queries of the same entity come next time. This also applies to the topic-level.

Figure 7 shows an example overall system architecture 700. A typical user action flow 702 is shown in a keyword search scenario. The user may search 704 by typing keywords and then follow up with some behavioural events 706, e.g., browsing the candidate list, clicking one of the candidates, browsing the content, and completing a transaction after a checkout. These events are streamed through the user events streaming module 708, which may be

implemented using Kafka streams. The events are saved in Data Lake 710 which is a data repository for storing data, e.g., BigQuery, or Presto.

Multi-level intention mapping 718

Figure 8 shows the expanded details of the Multi-level Intention Mapping 718 in Figure 7. It comprises keyword preprocessing module 802, keyword-level 804, entity-level 806 and topic-level 808 user intention mapping modules.

Keyword Preprocessing 802

The raw searched keywords are firstly corrected by the keyword correction unit 810 for spelling mistakes, as shown in Example 1.

Example 1: query “Macdonlad’s” → “Macdonald’s”

The keywords are then passed through an auto-completion unit 812, as shown in Example 2.

Example 2: query “Macdon” → “Macdonald’s”

The keyword correction unit 810 and completion unit 812, can be implemented using statistical methods to predict the most likely (corrected/auto-completed) words based on a corpus, e.g., the historical query logs. For example, the keyword correction unit can be implemented based on the Bayesian theorem (<https://norvig.com/spell-correct.html>) and the auto-completion unit can be implemented using NearestCompletion, MostPopularCompletion or HybridCompletion algorithms (Ziv Bar-Yossef and Naama Kraus, “Context-sensitive query auto-completion”, In WWW. 2011). These preprocessing units can also be implemented in the user search flow 702 as a search assistance/guidance, in which case the obtained keywords are already corrected and auto completed.

Keyword-level User Intention Mapping 804

Keyword-level User Intention Mapping 804 is the most granular level of user intention mapping. The searched keywords are passed through a basic text processing unit 814 which

- Removes the beginning and/or trailing space(s)

- Uncapitalizes the letters
- Replaces the single or multiple space(s) between words by “_”

An example pseudo-code is given below:

```
func convertToKeywordID(keywords string) string {
    s := strings.Trim(keywords, " ")           // step 1
    s = strings.ToLower(s)                     // step 2
    return regexp.MustCompile(`\s+`).ReplaceAllString(s, "_") // step 3
}
```

Keywords such as “thai food” and “Thai Food”, will be mapped to a unique keyword_id “thai_food”, so that in the downstream module, the behavioural events can be aggregated on the same basis. As shown in the diagram, the output of this module is the keyword_id.

Since each user may search every day, it may not be feasible to compute and store the behavioural signals for all the keyword_ids in the database (DB) 712 and feature store 714 due to increasing vocabulary volume. A set of the most frequent queries can occupy the majority of the search traffic. As shown in Figure 9 as example, the 7000 most frequent queries can account for >90% traffic for a specific food delivery use case. A keyword_id filter unit 816 performs some filtering logic based on the frequency. For example, the following logic can be implemented.

Take the most frequent keyword_ids up to a 90% threshold in a batch of queries.

Keyword_ids must on average appear more than a second frequency threshold: e.g, twice a day in the past 7 days. The filter unit 816 ensures that the keyword_id vocabulary is manageable in a practical situation and accounts for the majority of the traffic.

Entity-level User Intention Mapping 806

An Entity can be a word or a series of words, which consistently refer or refers to the same semantic entity. The user search queries usually contain entities. Let’s consider a few examples here.

Example 1: query = “Din Tai Fung at JEM”

Example 2: query = “Thai Food”

Example 3: query = “Thai Restaurant”

In Example 1, “Din Tai Fung” is a restaurant name corresponding to a merchant entity and JEM is a place of interest (POI) name corresponding to a POI entity. Examples 2 and 3 both refer to a cuisine name corresponding to a cuisine entity. Examples of different types of entity IDs may include:

- Merchant entity (so for example the name of the merchant e.g.: “Din Tai Fung”)
- Cuisine entity (so for example the name of the cuisine e.g.: “Thai”)
- Dish entity (so for example the name of the dish e.g.: “Chilli Crab”)
- POI entity (so for example the name of the location e.g.: “JEM”)

As a generalization to other applications, one can consider other entity types (e.g., Brand, Company, etc.) as long as they are suitable for that application.

Through the entity recognition unit 818, the specific entities can be extracted and assigned entity_id(s). For Example 2 and 3, it can tell that the user’s interest is in Thai cuisine and therefore both queries can be mapped to the same entity_id called “cuisine:thai”. By entity recognition, different queries with the same entity meaning can be grouped together as a unique entity_id such that the user behavioural signals can be aggregated at a general entity level.

The entity recognition unit 818, may be implemented, for example, according to Figure 10. The keywords (e.g., “Din Tai Fung At JEM”) are input into a sequence model 1002. The model will predict the output for each of the words indicating the tag the word belongs to. These tags are, for example,

B-Mer: beginning of a merchant entity

I-Mer: Inside a merchant entity

O: out of entity

B-Poi: beginning of a POI entity

...

In training, the tags are labelled beforehand for a training dataset and after the model is trained, such a model can be used in Entity Recognition module 818 to predict the entities based on the predicted keyword tags. In the above example, the model may predict “Din” -> B-Mer, “Tai” -> I-Mer, “Fung” -> I-Mer, resulting in “Din Tai Fung” as a predicted merchant entity and “JEM” as a predicted POI entity.

A query may contain multiple entities, as shown in Example 1. The system therefore includes an Entity Fusion/Pooling unit 820 after the Entity Recognition unit 818. The Entity Fusion unit 820 can perform concatenation of the recognised entities (or part of them based on the confidence score of each entity). The downstream behavioural signal aggregation will then be performed at the concatenated entity_ids. For Example 1, the user behaviours will be aggregated at the concatenated entity_ids of “merchant:din_tai_fung,poi:jem”. Alternatively, the Entity Pooling unit 820 performs pooling of the recognised entities based on their confidence score. For Example 1, if the confidence score for “merchant:din_tai_fung” is higher than “poi:jem” (as the user’s main intention is the merchant), “merchant:din_tai_fung” will be taken as the entity_id.

Topic-level User Intention Mapping 808

The highest level user intention mapping is topic-level. The keywords are firstly tokenized through a tokenization unit 822, followed by an embedding computation unit 824. The output of the embedding computation unit 824 is a dense vector (or so-called embedding) representing the textual information in the extracted keywords. Similar queries would have similar embeddings, which can be determined using a similarity measure. An example of a similarity measure is cosine similarity. The embeddings are then passed through a Bucketization/Clustering unit 826 (e.g., using K-means). Similar embeddings can be grouped to a bucket and after hashing 828, these buckets can be mapped to a topic_id 830.

Considering the below examples:

Example 1: query = “Seafood”

Example 2: query = “Long Beach Seafood”

Example 3: query = “Jumbo Seafood”

Example 4: query = “Pork Rib Soup”

Despite that Example 1 corresponds to a cuisine entity while Examples 2 and 3 correspond to some merchant entities, they are all related to a hidden topic of seafood. With the well-trained embedding computation unit 824, the dense vector of each will have a sufficient similarity measure to be clustered into a “seafood” topic bucket and finally mapped to one topic_id. Example 4 in this case will be mapped to another topic_id.

To implement this topic recognition functionality, an example is shown in Figure 11. The input keywords are passed through the tokenization 822, which is trained separately using a corpus (e.g., the historical query logs, or a collection of merchant menus, etc.). One can use, for example, SentencePiece to train such a tokenizer. The tokens (e.g., “jum”, “-bo”, etc.) are then passed through an embedding computation unit 824 (which may employ an LSTM or textCNN model) to compute an embedding for this query. This embedding computation unit 824 can be trained through a learning task, e.g., a topic classification task with the manually tagged topics as labels. It can also be trained through a CTR model where the user clicks/orders can be implicitly used as labels. The queries resulting in the click and/or order of the same merchant will hence be trained to have similar dense vectors. Essentially, the embedding computation unit is trained offline with the learning task. After the training is done, one can discard the learning task and just take out the embedding computation unit. Both tokenization and embedding computation units will then be plugged into 808 to serve as a functionality to convert keywords to embeddings.

Multi-level User Behavioural Signals Aggregation 716

Once the system has the user_id together with what has been obtained from 804, 806, and 808, the following values are obtained:

user_id
keyword_id
entity_id
topic_id

The behavioural events can then be aggregated on these keys to obtain behavioural signals. Except for the `user_id`, the other three aggregation keys involve multiple users such that the behaviours are aggregated among users under a specific search context (e.g., a `keyword_id`, an `entity_id`, etc.). This is beneficial because for a specific search context, a common user preference signal can be captured.

Behaviour categories may include but not be limited to:

- view behaviours
- click behaviours
- order behaviours
- dwell time

The view, click and order behaviours refer to the user view, click and order events. The dwell time refers to the user dwell time on a particular candidate (e.g., a menu page in food delivery use case after the user clicks one of the restaurants in the search results list). The behaviours chosen, will depend on the requirements of the application. The behaviours may also be adapted depending on the requirements of the application.

As shown in Figure 12, the user behavioural signal aggregation 720 and 716 is defined as a series of mathematical computations for the user events based on some aggregation keys. The first step is the GroupBy action 1202, which can be based on a single key (e.g., `keyword_id`), or composite keys (e.g., `keyword_id + candidate_id`). The aggregation 1204 can then be performed to obtain some intermediate signals, e.g., sum of clicks for a pair of composite keys. The last step is to perform some further processing 1206 based on one or multiple intermediate signals to derive the final behavioural signals, e.g., dividing “sum of clicks” by “sum of views” to derive a click through rate (CTR)-like signal.

Table 1 shows an example of the events data to be aggregated, where the keys are obtained from 718 and events are obtained from 708 and 710 in Figure 7. For this example, if both `keyword_id` and `candidate_id` are chosen in a combination to aggregate on, and it is desired to derive a signal called “`keyword_candidate_ctr`”, this can result in an example such as shown in Table 2. For the composite key `keyword_candidate_id = “thai_abcd”`, the signal

value of “keyword_candidate_ctr” is derived by dividing “sum of clicks” (=1) by “sum of views” (=2) taken from the first 2 rows in Table 1 where keyword_id = “thai” and candidate_id = “abcd”, which results in 0.5 in Table 2.

Table. 1 An example of source data to be aggregated

Keys			Events		
keyword_id	candidate_id	...	Views	clicks	...
Thai	abcd	...	1	0	...
Thai	abcd	...	1	1	...
Western	efgh	...	1	0	...
...	...	...	...	...	...

Table. 2 An example of behavioural signals (aggregated on keyword_id and candidate_id)

keyword_candidate_id	keyword_candidate_ctr	...
thai_abcd	0.5	
...	...	

Table. 3 Examples of behavioural signals

Category	Aggregation keys	Example behavioural signal	Description
click behaviours	user_id, candidate_id	user_candidate_ctr	Number of clicks / number of views for a user-candidate pair
	keyword_id, candidate_id	keyword_candidate_ctr	Number of clicks / number of views for a keyword-candidate pair
	keyword_id, candidate_id	keyword_candidate_click_perc	Number of clicks for a particular keyword-candidate pair / number of clicks under this keyword
	entity_id, candidate_id	entity_candidate_ctr	Number of clicks / number of views for an entity-candidate pair
	topic_id, candidate_id	topic_candidate_ctr	Number of clicks / number of views for a topic-candidate pair
order behaviours	user_id	user_avg_order_value	Average order value for a user
	keyword_id	keyword_checkout_candidate_pos_avg	Average of the checkout candidate position under this keyword
	keyword_id	keyword_checkout_candidate_pos_std	Standard deviation of the checkout candidate position under this keyword

	keyword_id, candidate_id	keyword_candidate_ctcvr	Number of orders / number of views for a keyword-candidate pair
	entity_id, candidate_id,	entity_candidate_order_perc	Number of orders for a particular entity-candidate pair / number of order under this entity
	topic_id	topic_checkout_prob	Number of orders / number of searches for this topic
dwell time	keyword_id, candidate_id	keyword_candidate_dwell_time	Dwell time for a keyword-candidate pair
	entity_id, candidate_id	entity_candidate_dwell_time	Dwell time for an entity-candidate pair

Table 3 shows more examples of user behavioural signals that can be derived through the aggregation. Each individual user_id, keyword_id, entity_id and topic_id can be used as a single aggregation key. For example,

“keyword_checkout_candidate_pos_avg” and “keyword_checkout_candidate_pos_std”

correspond to the aggregations performed on the keyword_id, and reflect the determinedness of a query. For example, the search query “mcdonald’s” would have higher determinedness than the search query “burgers” because the former one implies the user is determined for a particular restaurant. Since the best matched candidate, Mcdonald’s restaurant, would usually appear in the top after the search of “mcdonald’s” and users usually checkout from there, one would have $kw_checkout_mex_rank_avg_7d \approx 1$ and $kw_checkout_mex_rank_std_7d \approx 0$. On the other hand, users would checkout from different burger restaurants (i.e., different positions in the search result list) after the search of “burgers”, which will result in $kw_checkout_mex_rank_avg_7d > 1$ and $kw_checkout_mex_rank_std_7d > 0$. Each key can also be used together with the candidate_id to form a composite aggregation key. For example, the “keyword_candidate_click_perc” reflects the click popularity of a particular candidate compared to other candidates under the same searched query. As an example, it may be found that such a signal has a higher value for “seafood_candidate-1” than “seafood_candidate-2” as “candidate-1” is more popular with users who searched “seafood” previously.

Deriving behavioural signals can be done offline or in a real time manner. This may be important for an online search and ranking system where immediate user behavioural patterns are important and need to be captured.

Feature Pipeline

The feature pipeline 720 and 716 generates the behavioural signals on a regular basis (e.g., daily) or a real time manner. For the offline behavioural signals, they can be implemented as a scheduled job. For example, the aggregation logic can be scheduled and run every day using a workflow scheduler such as Airflow. The real time behavioural signals can be implemented as a stream processing job, for example, using Flink.

Both jobs may save the signals into relational databases 722 and 712, for example, BigQuery or Presto. The behavioural signals may be saved as tables (e.g., like Table 4) where the index will be the aggregation keys. Separate tables can be used for different kinds of aggregation key, e.g., one table for signals aggregated on keyword_candidate_ids and another table for signals aggregated on entity_candidate_ids. Date partitions can be used inside each table, so that the signals computed at a particular day will be saved in a corresponding partition. By doing this, behavioural signals are stored across multiple past days, which will be used for model training.

Table. 4 An example table in relational database to store behavioural signals for model training

Index	Behavioural Signals		Partition
keyword_candidate_ids	keyword_candidate_ctr	...	date
thai_abcd	0.5	...	2021-10-01
thai_efgh	0.3	...	2021-10-01
...	...	...	...

The signals may also be saved in a feature store 714 for model serving. Such a feature store is essentially a database but with low latency such that the stored features and/or behavioural signals can be fetched rapidly and used by the online model. Some exemplary feature stores include DynamoDB and FitreStore. The feature store is usually implemented

as a non-relational database, e.g., as shown in Figure 13. There may be no date partition for the feature store 718, and the feature pipeline will use the latest aggregation results to either overwrite the signal values for the existing ids or append new records for the new ids.

Model Training 724

Figure 14 shows how the user behavioural signals 1402 can be used for model training 724. The behavioural signals 1402 can be used alone, or in combination with other features, e.g., user features, candidate features, context features (time of day, day of week, etc.) or any other features. Similar to a typical ML application, the features are passed through a feature engineering stage 1410 where one can perform, e.g., bucketization of numerical features, one-hot encoding of categorical features, feature crossing, embedding, etc. To train the ranking model 1412, for example, the Wide & Deep model architecture can be used. The L2R framework can also be adopted to formulate a list-wise loss function to train such a model.

The trained model may be served online 726 where the user behavioural signals, together with other user features, may be fetched from the feature store 718 using the keys.

It will be appreciated that the invention has been described by way of example only. Various modifications may be made to the techniques described herein without departing from the spirit and scope of the appended claims. The disclosed techniques comprise techniques which may be provided in a stand-alone manner, or in combination with one another. Therefore, features described with respect to one technique may also be presented in combination with another technique.

Claims

1. A communication server apparatus for ranking search results, the communication server comprising a processor and a memory, the communication server apparatus being configured, under control of the processor, to execute instructions stored in the memory, to:
 - store a plurality of sets of search intent identifiers, wherein respective sets of search intent identifiers are associated with different respective levels of semantic granularity, and wherein a first set of the search intent identifiers comprises a set of keyword IDs;
 - receive a user search request;
 - computationally map the user search request to one or more estimated keywords, the estimated keywords selected from the set of keyword IDs;
 - computationally map the one or more estimated keywords to one or more estimated first derived features, the estimated first derived features selected from a second set of the search intent identifiers that is not the set of keyword IDs;
 - determine one or more behavioural signals based on the one or more estimated keywords and/or the one or more estimated first derived features; and
 - rank a set of search results based on the user search request and the one or more behavioural signals.
2. The server apparatus of claim 1, wherein the second set of search intent identifiers comprises a set of entity IDs, and the estimated first derived features are one or more estimated entities.
3. The server apparatus of any preceding claim, further configured to determine or extract from the estimated keywords, one or more estimated second derived features selected from a third set of the search intent identifiers that is not the set of keyword IDs.
4. The server apparatus of claim 3, wherein the third set of the search intent identifiers comprises a set of topic IDs, and the second derived features are one or more estimated topics.

5. The server apparatus of claim 3 or 4, wherein the one or more behavioural signals are further determined based on the estimated second derived features.
6. The server apparatus of any one of the preceding claims, wherein the one or more behavioural signals are also determined based on context features, candidate features, and any combination thereof.
7. The server apparatus of any one of the preceding claims, wherein the one or more behavioural signals are also determined based on one or more of: view behaviours, click behaviours, order behaviours, and dwell time.
8. The server apparatus of claim 7, wherein the view behaviours are determined based on the number of times a user views an internet page and one or more corresponding search intent identifiers associated with the respective internet page.
9. The server apparatus of claim 7 or 8, wherein the click behaviours are determined based on the number of times a user clicks to access an internet page and one or more corresponding search intent identifiers associated with the respective internet page.
10. The server apparatus of any of claims 7 to 9, wherein the order behaviours are determined based on transaction parameters in relation to a user and an internet page and one or more corresponding search intent identifiers associated with the respective internet page.
11. The server apparatus of any of claims 7 to 10, wherein the order behaviours are determined based on a dwell time in relation to a user and an internet page and one or more corresponding search intent identifiers associated with the respective internet page.
12. The server apparatus of any preceding claim, further configured to update the set of keyword IDs based on candidate keywords used in a plurality of users' search requests over a predetermined time period, and wherein only the candidate keywords having a frequency above a pre-determined threshold, are used to update the set of keyword IDs.

13. The server apparatus of claim 5, further configured to parse the user search request into a series of tokens.

14. The server apparatus of claim 13, further configured to generate an embedding of the series of tokens.

15. The server apparatus of claim 14, further configured to cluster and map the embedding to the estimated second derived features.

16. A method performed in a communication server apparatus for ranking search results, the method comprising, under control of a processor of the communication server apparatus:

- storing a plurality of sets of search intent identifiers, wherein respective sets of search intent identifiers are associated with different respective levels of semantic granularity, and wherein a first set of the search intent identifiers comprises a set of keyword IDs;

- receiving a user search request;

- computationally mapping the user search request to one or more estimated keywords, the estimated keywords selected from the set of keyword IDs;

- computationally mapping the one or more estimated keywords to one or more estimated first derived features, the estimated first derived features selected from a second set of the search intent identifiers that is not the set of keyword IDs;

- determining one or more behavioural signals based on the one or more estimated keywords and/or the one or more estimated first derived features; and

- ranking a set of search results based on the user search request, and the one or more behavioural signals.

17. A computer program or computer program product comprising instructions for implementing the method of claim 16.

18. A non-transitory storage medium, storing instructions, which when executed by a processor, causes the processor to perform the method of claim 16.

19. A user communication device for communicating with a communication server, the communication device comprising a processor and a memory, the communication device being configured, under control of the processor, to execute instructions stored in the memory to:

send a search request from the user to a server; and

receive a ranked set of search results from the server, wherein the ranking is based on the user's search request, and one or more behavioural signals, the one or more behavioural signals determined from one or more estimated keywords computationally mapped from the user's search request, and one or more estimated first derived features computationally mapped from the one or more estimated keywords.

20. A communication system, comprising:

a communication server;

at least one merchant communication device;

at least one user communication device; and

communication network equipment configured to establish communication with the communications server, the at least one merchant communication device, and the at least one user communication device;

wherein the user communication device comprises a first processor and a first memory, the user communications device being configured, under control of the first processor, to execute first instructions stored in the first memory to:

request the communication server provide search results;

and wherein the communication server comprises a second processor and a second memory, the communication server being configured, under control of the second processor, to execute second instructions stored in the second memory to:

store a plurality of sets of search intent identifiers, wherein respective sets of search intent identifiers are associated with different respective levels of semantic granularity, and wherein a first set of the search intent identifiers comprises a set of keyword IDs;

receive a user search request;
computationally map the user search request to one or more estimated keywords, the estimated keywords selected from the set of keyword IDs;
computationally map the one or more estimated keywords to one or more estimated first derived features, the estimated first derived features selected from a second set of the search intent identifiers that is not the set of keyword IDs;
determining one or more behavioural signals based on the one or more estimated keywords and/or the one or more estimated first derived features;
and
rank a set of search results based on the user search request, and the one or more behavioural signals;
and wherein the merchant communication device comprises a third processor and a third memory, the merchant communication device being configured, under control of the third processor, to execute first instructions stored in the third memory to:
receive data regarding a transaction in relation to the ranked set of search results from the communication server.

21. A method performed in a communication server apparatus for ranking search results, the method comprising, under control of a processor of the communication server apparatus:

computationally mapping a user search request to one or more estimated keywords;
computationally mapping the one or more estimated keywords to one or more estimated entities;
parsing the user search request into a series of tokens;
generate an embedding of the series of tokens;
clustering and mapping the embeddings to one or more estimated topics; and
ranking a set of search results based on the user search request, the estimated keywords, the estimated entities and/or the estimated topics.

22. A computer program or computer program product, comprising instructions for implementing the method of claim 21.

23. A non-transitory storage medium, storing instructions, which when executed by a processor, causes the processor to perform the method of claim 21.