Understanding Search Queries in Natural Language

Zuzana Nevěřilová[1,2] and Matej Kvaššay[1]

[1] Konica Minolta Laboratory Europe, Brno
[2] NLP Centre, Masaryk University, Brno
{zuzana.neverilova,matej.kvassay}@konicaminolta.cz

Abstract. This work is part of a project aiming to provide one single search endpoint for all company data. We present a search query parser that takes a speech-to-text output, i.e. a sentence. The output is a structured representation of the search query from which a SPARQL query is generated. The SPARQL is then applied to an ontology with the company data.
The parsing procedure consists of two steps. First, the search intent is detected, second, the query is parsed based on the search intent. For the intent classification, we use word embeddings with boosting of top 5 words, and support vector machines. For the parsing, we use semantic role labeling, named entity recognition, and external resources such as ConceptNet and DBPedia. The final parsing step is rule-based and related to the ontology structure.
The intent classifier accuracy is 94%. In the subsequent manual evaluation, the resulting structures were complete and correct in 51% cases, in 34.57% of cases it was complete and correct but it also contained irrelevant information.

Keywords: search intent; search query parsing

1 Introduction

Search in corporate data is one of the pain points for many people working in the office. Apart from searching physical objects, they also look for digital objects. This task is considered to be annoying and surprisingly difficult. In our tool, we aggregate all possible data sources the company is working with, such as company wiki, emails, task management tool, employee profiles, or instant messages, so a powerful search engine is a must-have. The user interface allows among other voice inputs. The voice signal is transcribed into text and the system has to interpret this text into a search query. Such inputs are completely different from the common search queries which are mostly keyword-based.

In case of search queries in natural language, we have to parse and interpret a short text, consisting mostly of one or two sentences with many entities and named entities. Often the sentence expresses relationships between the entities. Sometimes, implicit knowledge has to be added to the interpretation.

Aleš Horák, Pavel Rychlý, Adam Rambousek (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2018, pp. 85–93, 2018.

The system has three main components: one detects the main search intent (e.g. a file or a person), the other parses the query into a structure from which a SPARQL query is constructed. In the last component, the search results are ranked and presented to the user from the highest rank.

1.1 Paper Outline

In Section 2, we describe in short the current aspects of search engines and the current search query parser. Section 3 focuses on methods we have used, particularly on . Section 4 discusses the evaluation criteria. Section 5 contains final remarks.

2 Related Work

Search platforms are nothing new. Even in the open source world, one can find search engines combining full text search with search query parsing, faceted search (interactive filtering), synonym expansion, and other features. One such platform is Open Semantic Search [3].

Most search engines are keyword-based fulltext (such as search in the Web) or faceted-based (such as search in a library) or combination of both. In addition, search engine can provide instant feedback or clarification dialogue, and thus, in such cases it becomes something between a search engine and a question answering system. The dialog-like search is also present in personal assistants such as Siri, Cortana, Alexa, and other where the interface is spoken.

The presentation of search results is not a simple list of items anymore. Even web search engines try to guess the user search intent and in some cases provide the direct answer. For example, Google Search provides a calculation if the user enters a mathematical formula, a conversion if the user enters query such as "15 EUR in CZK", or a description if the user enters a named entity (e.g. for "Marylin Monroe" it returns "a film actress"). Many aspects of search query presentation are described in [4].

In the current version of our project, the search engine provides faceted search as shown in Figure 1 and sentence query search intended to work together with voice input. Users can also write search sentences into the search input box but in reality, nobody expects them to do so. For parsing the search sentence, we use Google DialogFlow [4] with predefined dialog intents. The output of sentence parsing by DialogFlow is a structured object such as a Python `dict`, example of such output can be seen in Figure 2.

The granularity of intents is quite high, e.g. searching documents shared with someone is different intent than searching documents from a meeting. DialogFlow is provided by 10–30 example queries and creates a generalization for this particular intent.

[3] `http://opensemanticsearch.org`

[4] `http://dialogflow.com`

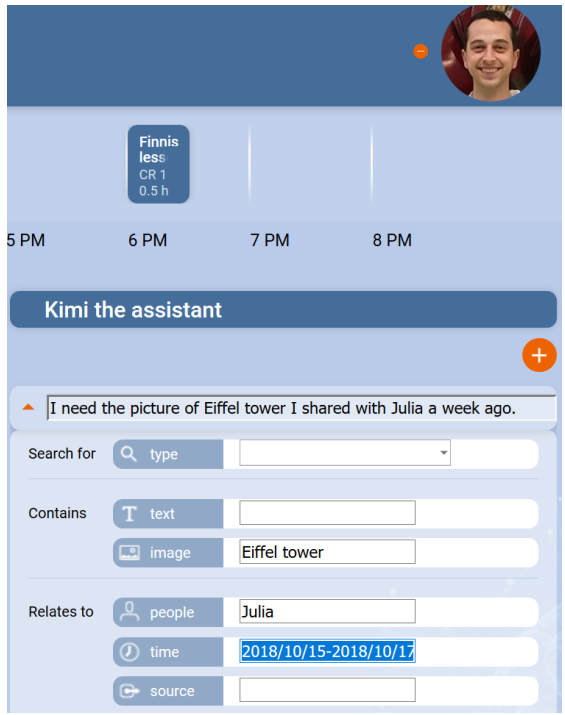

Fig. 1: Search input box and faceted search presented in the frontend

```
{
 ...
 "parameters": {
  "openingPhrase": "find",
  "givenName": "Bob",
  "lastName": "",
  "document": {
    "type": "presentation",
    "topic": "artificial intelligence"
 },
  ...
  "score": 0.9966866513437793
}
```

Fig. 2: Sample output for input “Find the presentation about artificial intelligence that Bob sent to me”.

It seems that DialogFlow extensibility is limited: every time a new intent is added to the current ones, the confidence numbers for all intents decrease.

The aim of this work is to provide a search query parser with at least the same accuracy as DialogFlow on transcribed sentences and higher extensibility.

3 Methods

The design of the search query parser consists of two basic modules: intent classifier and search query parser adapted to a particular intent. For example, the prepositional phrase beginning with "in" means usually a location. In case of digital objects, this location is digital as well (for example, "in office" means "in the Office application"), while in case of physical objects, the location is also physical (for example, "in office" means "in somebody's office").

3.1 Intent Classifier

Problem definition According to an internal survey, people working in office most frequently look for text documents (21% cases), persons (10% cases), multimedia files (7.8% cases), personal belongings, emails (6.4% cases), web pages, locations, tasks, presentations and others with lower frequency. Based on this survey, we identified six classes of search intents, described in Section 4.

Proposed model To classify intents, all tokens of the query are lower-cased and stop-words are removed. Tokens are mapped to 300-dimensional word embeddings using publicly available vocabulary of FastText [2] vectors trained on CommonCrawl dataset. Missing words are ignored. Vectors are then aggregated by averaging in two ways:

1. Average of vectors of first k words of the query
2. Average vectors of all words of the query

Both vectors are then concatenated into single 600-dimensional representation for each sample. Motivation for this **double representation** is our observation that natural language search query often contains most informational words for intent classification at the beginning of the query sentence (e.g. word "document" in "look for a document which contains image of elephant"). By averaging words from beginning of the sentence, we encode information about beginning of the search phrase and let the classifier exploit this positional-specific information. This simple approach enables the classifier to focus on specific parts of the query. The trade-off is the increase in feature vector dimension.

We used SVM [1] classifier with RBF kernel that is also able to evaluate confidence of the prediction.

3.2 Search Query Parser

We assume the search query is composed of one or a few sentences. First, we apply semantic role labeling to each sentence, then we apply rules to parse each of the arguments. The rules depend on the search intent.

Semantic Role Labeling Semantic role labeling (SRL) decomposes each clause of a sentence to predicate-argument structure. Historically, SRL used syntactic parsing, however, the Deep SRL [3] which is based on neural networks outperforms the previous approaches.

We use Deep SRL as a server that for a given sentence outputs separate clauses. For each clause, it outputs the predicate and its arguments. The arguments are the same as in PropBank[5]: numbered arguments `ARG0`–`ARG5`, and predicate and phrasal modifiers (e.g `ARGM-LOC` for locations or `ARGM-TMP` for temporals). We do not consider PropBank links.

Rule-based Argument Parsing Each numbered argument is rule-based parsed. In future, we consider to induce the rules from the ontology scheme but in the current version, the connection with ontology is very limited. We treat predicate and phrasal modifiers, and numbered arguments in different ways.

Predicate and phrasal modifiers For arguments of type `ARGM-LOC` or `ARGM-TMP`, the parsing is straightforward: we consider the whole content of the argument as one unit of the same type as the argument (e.g. location or time).

Argument containing the main intent We parse the argument that contains the main intent in a different than the other numbered arguments. The main intent is always the syntactic head of the argument (if is not, the parsing cannot continue). All dependent components are modifiers of the intent. For example, if the argument contains "pdf file", the main intent is "file" and "pdf" is a constraint to file format.

Arguments not containing the main intent Other numbered arguments are processed together with the predicate since the predicates describe relations (e.g. contain, create, share, ...) between the main intent and other entities. If the extracted entities are recognized as potential objects in the graph database (such as users), they have to have a relationship to the main intent or other objects. In other cases, the entities are identified as keywords. We use SpaCy[6] with large English model for tagging and recognizing named entities.

The overall result of the parsing is a structure. An example can be seen in Figure 3. In the output structure, we consider only autosemantic tokens, however, other part of speech can modify the relation. For example, if the query contains a named entity "Bob", it can be interpreted as the owner or creator

[5] `https://propbank.github.io/`

[6] `https://spacy.io`

```
{
...
 "tokens": [
    {"text": "document",
     "relation": {"value": "intent", "confidence": 0.5},
     "label": {"value": "presentation", "confidence": 1.0}
    },
    {"synonym": [
        {"value": "AI", "confidence": 0.5},
        {"value": "ai", "confidence": 0.5}
     ],
     "informationSource": "conceptnet",
     "text": "artificial intelligence",
     "relation": {"value": "keyword", "confidence": 0.5}
    },
    {"text": "Jane Smart",
     "entity": {"value": "PERSON", "confidence": 0.5},
     "relation": {"value": "sharedWithPERSON", "confidence": 0.5}
    }
  ]
}
```


Fig. 3: Sample output for input "Find the document about artificial intelligence that Jane Smart provided to me.".

of a document (e.g. "find documents by Bob") but it can be also interpreted as a keyword in the document (e.g. "find documents about Bob"). Multiword expressions are identified and treated as a single token. Foreach token, the *relation* is determined. The token text and token relations are necessary, since the resulting structure is later converted into a SPARQL query in the form of triples $(mainintent, relation, label)$.

We process multiword expressions, using syntactic constraints (NOUN-NOUN, ADJ-NOUN, PROPN-PROPN) and external resources. Particularly, we use ConceptNet[7] and DBPedia[8] to confirm that a multiword expression candidate is a single meaning unit. In most cases, the greedy approach (preferring "team building" over "team" and "building" which all three exist in external resources) is the best. In addition, ConceptNet provides synonyms that can later be used to expand the SPARQL query.

4 Evaluation

We evaluated the two parts of the system separately.

[7] `http://conceptnet.io/`
[8] `https://wiki.dbpedia.org/lookup`

4.1 Query intent classification

For evaluation we used internally created data-set of 441 query examples from 6 categories:

- Calendar event (cal)
- Message (msg) – email or instant message
- Multimedia content (mul) – image, video, or audio files
- Personal information (per)
- Task (tsk) – task description optionally with an assignee and a due date
- Text document/spreadsheet (txt)

The data were split 50/50 into category-balanced sets of 220 examples for training and 221 examples for testing due to small available sample size. We evaluated proposed query intent classifier with double representation and ad-hoc selected $k = 5$ against a baseline model, which was the same model without additional average vector for 5 first tokens.

Table 1: Test set performance model with single (word vector average, dim=300) and double (word vector average + first 5 words vector verage,dim=600) representation.

	single representation				double representation			
Category	Precision	Recall	F-1	Support	Precision	Recall	F-1	Support
cal	0.94	0.83	0.88	36	1.00	0.94	0.97	36
msg	0.80	0.75	0.77	32	0.90	0.84	0.87	32
mul	0.80	0.62	0.70	26	1.00	0.96	0.98	26
per	0.93	0.88	0.90	48	0.94	0.96	0.95	48
tsk	1.00	0.90	0.95	10	1.00	1.00	1.00	10
txt	0.74	0.91	0.82	69	0.90	0.96	0.93	69
micro avg	0.83	0.83	0.83	221	0.94	0.94	0.94	221
macro avg	0.87	0.81	0.84	221	0.96	0.94	0.95	221
weighted avg	**0.84**	**0.83**	**0.83**	221	**0.94**	**0.94**	**0.94**	221

In the test set classification performance results (Table 1), we show that the model with double representation achieves high average precision and recall (≥ 0.94) on the test set. We have also shown that adding average vector of first 5 tokens to all word vector average improved the results by at least 10% compared to baseline. The confusion matrices are presented in Figure 4.

4.2 Evaluation of the Search Query Parser

A crucial question for evaluation is whether the parsed structure can be transformed to a SPARQL query that returns correct results. We realized that

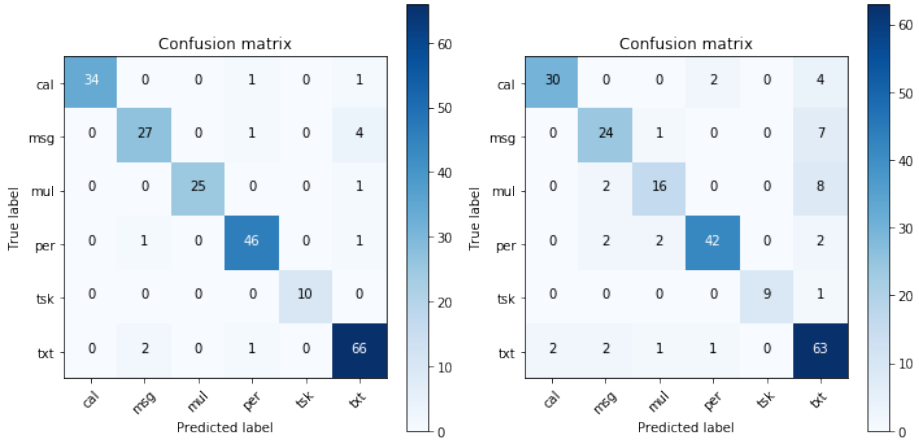

Fig. 4: Test set confusion matrix for model without double representation: word vector average + first 5 words vector average, dim=600 (left) and word vector average, dim=300 (right)

some ambiguity is present even in search queries, e.g. a presentation can be a video, an event, or a PDF/PPTX file. We therefore considered the parsing to be correct if it returned one meaningful interpretation of the sentence. We also wanted that all relevant parts of the sentence were considered in the parsed structure. The relevance was judged using a common sense interpretation of the sentence.

We evaluated manually the parsing on 80 example search sentences. 41 sentences were parsed completely and correctly. In 11 sentences, a relevant token was not recognized. In 3 cases out of these 11, it was a related person, in the remaining cases, it was a keyword. In 28 cases, an irrelevant token was extracted and included in the output structure. This was a case in sentences such as “find an AI expert in the London office” where the word “office” is not relevant for the search. In 6 cases, the relation was detected incorrectly.

5 Conclusion and Future Work

We proposed a natural language search query intent classification model with double representation allowing classifier to focus on specific part of the query and we have shown that this representation can significantly increase the classifier performance in experimental setting.

The sentence parser benefits from the intent classification, and uses semantic role labeling. Parsing of each argument is rule based. Even though we evaluated the parser on a limited number of sentences, we can see that its recall is plausible.

Possible improvements of the query intent classification model include using shorter word embeddings and more granular split of queries to include more

position-specific information into the feature vector and usage of domain-specific word embeddings additionally to embeddings model trained on public data-set.

We also plan to tie the parser more closely to the ontology scheme. The ideal situation would be a parser that can adapt on the ontology scheme modifications.

Acknowledgements. This research was supported by Konica Minolta Laboratory Europe. This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin OP VVV project CZ.02.1.01/0.0/0.0/16_013/0001781.

References

1. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (Sep 1995)
2. Grave, E., Mikolov, T., Joulin, A., Bojanowski, P.: Bag of tricks for efficient text classification. In: In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics. pp. 427—-431 (2017), http://arxiv.org/abs/1607.01759
3. He, L., Lee, K., Lewis, M., Zettlemoyer, L.: Deep Semantic Role Labeling: What Works and What's Next. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics (2017)
4. Tunkelang, D.: Search Results Presentation (Feb 2018), https://queryunderstanding.com/search-results-presentation-7d6c6c384ec1