Query Intent Detection using Convolutional Neural Networks

Homa B. Hashemi
Intelligent Systems Program
University of Pittsburgh
hashemi@cs.pitt.edu

Amir Asiaee, Reiner Kraft
Yahoo! inc
Sunnyvale, CA
{amirasiaee, reiner}@yahoo-inc.com

ABSTRACT

Understanding query intent helps modern search engines to improve search results as well as to display instant answers to the user. In this work, we introduce an accurate query classification method to detect the intent of a user search query. We propose using convolutional neural networks (CNN) to extract query vector representations as the features for the query classification. In this model, queries are represented as vectors so that semantically similar queries can be captured by embedding them into a vector space. Experimental results show that the proposed method can effectively detect intents of queries with higher precision and recall compared to current methods.

Keywords

Query Understanding, Convolutional Neural Networks, Query Vector Representation

1. INTRODUCTION

Modern search engines go beyond retrieving relevant documents. To satisfy a user's information needs, they also aim to display a concise answer to the user's query. Therefore, they require a deep understanding of the user's query. Identifying the intent behind the query is a crucial step toward reaching this goal. This not only helps display semantically enriched search results [2], but also helps improve ranking results by triggering a vertical search engine in a certain domain (e.g., image search, news search, and job search engines) [7]. For example, the query "United States president", which asks for factual information, would return *Barack Obama* as the answer, and the query "Pittsburgh pics" will show the results of the image search engine first.

Query intent detection is a challenging task because queries are usually short, and identifying the exact intent of the user requires more context beyond the keywords. Moreover, the number of intent categories could be very high. Most approaches require a huge amount of human effort to meet these challenges, either by defining patterns for each intent class or by defining discriminative features for queries to run statistical models. Instead, we propose a statistical method that automatically extracts discriminative features for queries.

Our approach is to use deep learning to find *query vector representations*; then use them as features to classify queries by intent. One of the advantages of using query vector representations is to embed queries in a space so that semantically similar queries are close to each other. For example, the queries "Michelle Obama age" and "Barack Obama wife birthday" will be close to each other; thus, they may be assigned the same intent.

Deep learning is mostly used in text processing tasks through leveraging *word vector representations* (such as `word2vec` [11]) which are essentially vectors with encoded semantic features of words in their dimensions. In this paper, we learn *query vector representations* which utilize convolutional neural networks (CNN) that is trained on top of *word vector representations*. CNN was originally invented for computer vision, and has recently been shown to be useful in many natural language processing and information retrieval tasks [3], such as semantic parsing [17], sentence modeling [8, 9], document ranking [16], document similarity [5] and query reformulation [12]. Our task differs from sentence embeddings in the sense that queries are often short and unstructural. It also differs from query embeddings in that it does not use various types of features, such as click-through data [16] or search sessions [12].

In this paper, we present our approach to identifying query intent as a multi-class classification task with extracted query vector representations as features. We show the effectiveness of this approach in different experimental settings. The results show that query vectors obtained from the pre-trained CNN model perform well on query intent detection and are comparable with bag-of-word features. Moreover, although they are learned automatically from the training data, they significantly outperform the carefully designed rule-based baseline.

2. RELATED WORK

Similar to most natural language processing tasks, there are two main approaches to identifying query intent: rule-based and statistical methods. The rule-based systems use predefined rules to match new queries to their intents. While these systems are usually precise (i.e, if they detect an intent for a query, it is correct most of the time), their coverage is low (i.e, they are not able to detect intents for many queries due to a shortage of rules). In addition, the rules need to be carefully engineered by human experts; designing a new rule for the system might take a few hours. Thus, scaling up these systems to a large number of intents is difficult and needs a huge amount of human effort.

Statistical models are another popular method to detecting intents either, with a supervised [10, 7] or unsupervised method [15,

Presented at WSDM QRUMS 2016 Workshop
© 2016 ACM. ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

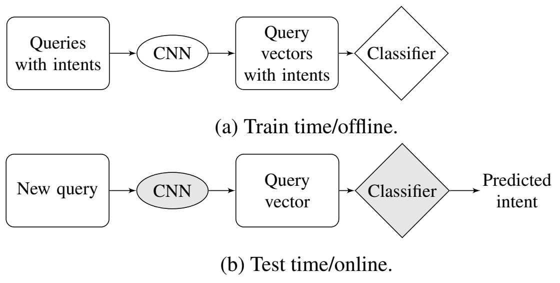

Figure 1: Model architecture in the train and test time. Shaded nodes means they are trained.

2, 14]. In a supervised method, a classifier is trained to use a set of queries with their intents by learning the parameters of the model. Given a new query, the classifier can assign an intent to it. However, one of the conditions of these methods is to design discriminative features. This task is called *feature engineering* and requires domain knowledge. Playing with the features will lead to the construction of state-of-the-art systems. Previous studies used various types of features, such as search sessions [1], click-through data [10], and Wikipedia concepts [7]. In this paper, instead of exploiting man-made classification features, we use the automatically generated query vector representations as the feature set.

3. QUERY INTENT IDENTIFICATION

In this section, we describe our proposed model and a set of contrastive systems that we used as baselines to detect query intent.

3.1 Our Method

Our proposed method is a classification-based approach, but instead of engineering query features, we use convolutional neural networks (CNN) to automatically extract query vectors as the features.

3.1.1 Model Architecture

Our proposed model has two main steps: 1) training the model parameters in the offline time, and 2) running the model over new queries in the online time. Figure 1 depicts these steps. During training, we utilize the labeled queries to learn the parameters of convolutional neural networks and the intent classifier. During run time, we pass the query through these two components to detect its intent.

3.1.2 Convolutional Neural Networks (CNN)

Given a query, the goal of the convolutional neural network component is to find its vector representation. We use the CNN architecture of Collobert et al. [3] and its variant implementation by Kim [9]. We slightly modified the model during the testing phase to get the query representations after the max pooling layer by removing the softmax layer. Having query vector representations will then help us to combine them with other query features such as language modeling. Figure 2 illustrates the CNN model during the run time.

The query representations are trained on top of pre-trained word vectors which are updated during CNN training. We use the publicly available `word2vec` [11][1] vectors that are trained on Google News. During the training phase, the parameters of the CNN model are learned by passing multiple filters over word vectors and then applying the max-over-time pooling operation to generate features

[1]`https://code.google.com/p/word2vec/`

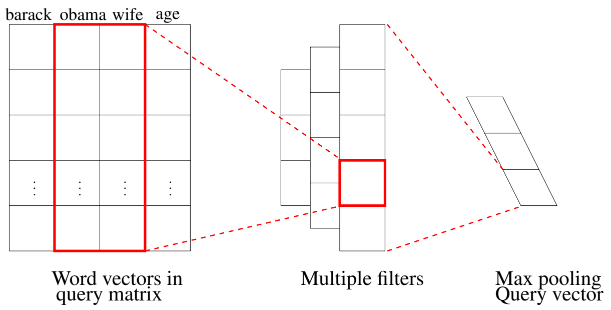

Figure 2: The CNN model for a query with four words. The network has filters with width 2, 3 and 4.

which are used in a fully connected softmax layer. Finally, we use the cross-entropy loss function for learning parameters of the model. Similar to Kim [9], we use dropout [6] to regularize the change of parameters by randomly setting some weights to zero that prevents overfitting.

3.2 Contrastive Methods

We compared our approach with both a rule-based model as well as a classification-based model which has features from word vectors or the language modeling of the sentence.

3.2.1 Rule-based system

The rule-based method is designed in a commercial search engine to label queries with their intents. It uses a set of hand crafted and automatically generated regular expressions. It also utilizes an entity tagger to map the query words to their applicable name entities. When queries have similar patterns, this approach works pretty well. For example, if the query is “Brad Pitt age” or “Barack Obama how old”, the intent will be recognized as a `person age` because these queries contain a person entity with special keywords related to birthdate. The keywords are mostly manually collected by experts. On the other hand, the rule-based system lacks scalability. When a new query appears that do not match any other previous patterns, the rule-based system is not able to assign an intent to it. Therefore, it misses the query even if the query is only a small variant of previous patterns.

3.2.2 Bag-of-words Features

As another baseline to detect query intent, we consider bag-of-words features in a classification-based setting. Language modeling is regarded as a strong baseline for most NLP applications. In this approach, we consider N-grams[2] in the query as features and run a classifier to learn parameters. Furthermore, the queries with exactly same words will have the high probability of having the same intent. However, this model ignores similar N-grams if they are not appeared in the training data.

3.2.3 Aggregated Word Vector Features

Aggregation of query word embeddings is another simple set of features. The goal is to find an embedding for a query and use it as a feature to train the intent classifier. Instead of passing query word vectors through a convolutional neural network, we can simply get the word vectors of each query word and sum them together (`Sum w2v`) or get their average (`Average w2v`). The resulting query embedding will have the same dimension of word vectors. In our experiments, we use `word2vec` word embeddings.

[2]We report unigram and bigram results as they performed better

Table 1: Query intent detection results with 125 low-level intent classes

Method	Accuracy(%)	Avg. Precision	Avg. Recall	Avg. F-score
Rule-based	75	0.46	0.39	0.40
Sum w2v	74.5	0.48	0.34	0.36
Average w2v	75.2	0.49	0.34	0.36
Unigram	81.3	0.56	0.45	0.47
Unigram+bigram	81.2	0.56	0.44	0.46
CNN	**81.6**	**0.62**	**0.47**	**0.50**

Table 2: Query intent detection results with 14 high-level intent classes

Method	Accuracy(%)	Avg. Precision	Avg. Recall	Avg. F-score
Rule-based	77	0.45	0.34	0.36
Sum w2v	86	0.49	0.24	0.28
Average w2v	85.9	0.49	0.25	0.29
Unigram	89.9	**0.71**	0.37	0.43
Unigram+bigram	89.9	0.68	0.34	0.40
CNN	**90.3**	0.56	**0.43**	**0.47**

Table 3: A sample of intent taxonomy in two levels

High-level intent	Low-level intent
Movie	Rating Cast Length Release date
Person	Spouse Birth date Children Height

Table 4: Statistics on the query intent data

# of queries	10,000
# of low-level intent classes	125
# of high-level intent classes	14

4. EXPERIMENTS

We conduct a series of experiments to test the usefulness of convolutional neural networks in query understanding.

4.1 Data

The queries are collected from a commercial search engine logs. To make the golden data set of queries with their intents, a sample of queries were manually annotated each day from January 2014. The query intents are based on the locally designed taxonomy with more than 3800 intents in the lowest level. The higher level of the taxonomy has 350 intent categories. Table 3 shows a sample of intent categories. In our experiment, we selected 10,000 queries that their gold labels were appeared at least 10 times in the annotated dataset. Some statistics of the data used in our experiments is shown in Table 4.

4.2 Experimental Setup

In our experiments, to do the classification, we run a 10-fold cross validation with the standard Random Forest (`n_estimators = 100`) in scikit-learn toolkit [13].[3]

In the CNN model, we use 3 filters with window sizes of 2, 3 and 4 with 100 feature maps each. These window sizes will capture bigram, trigram and 4-gram information of the query. We employ dropout for regularization with dropout rate of 0.5. We also use rectified linear units and mini-batches with size of 50.

We report accuracy and average precision, recall and Fscore as performance measures. The accuracy of a method is:

$$Accuracy = \frac{\text{\# of correctly predicted query intents}}{\text{Total \# of queries}} \quad (1)$$

The precision, recall and F-score are calculated for each intent class (either low-level or high-level) and then averaged to get the overall numbers. The precision and recall of an intent class i (e.g. movie rating) is calculates as:

$$Precision_i = \frac{\text{\# of correctly predicted intents with the intent } i}{\text{Total \# of predicted intents with the intent } i} \quad (2)$$

$$Recall_i = \frac{\text{\# of correctly predicted intents with the intent } i}{\text{Total \# of human annotated queries with the intent } i} \quad (3)$$

The F-score for the given intent class of i can then be calculated as harmonic mean of precision and recall:

$$F\text{-}score_i = 2 \times \frac{Precision_i \times Recall_i}{Precision_i + Recall_i} \quad (4)$$

4.3 Results and Discussion

To evaluate the usefulness of our proposed query detection method, we have conducted two classification and a clustering experiments. For the classification experiments, we validate the CNN model using standard classification measures and compare it with the contrastive baselines. For the clustering task, we use the extracted query representation as features to group queries.

4.3.1 Low-level intent detection

than higher-order N-grams in our experiments.

[3]We have also tried Gradient Boosting classifier and SVMs with LibLinear toolkit [4], but Random Forest learners obtained the best results in our multi-class classification task.

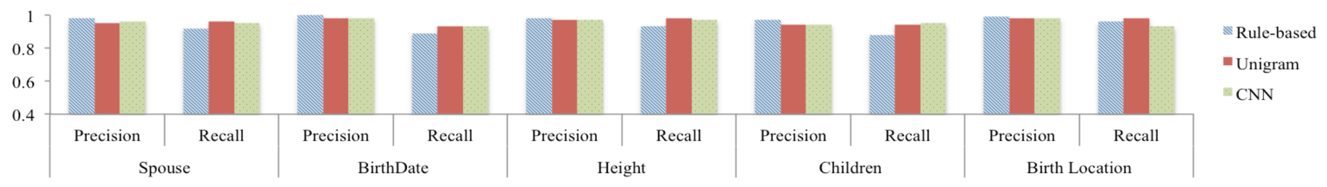

Figure 3: Intent detection results for the low-level intents of `Person` intent class.

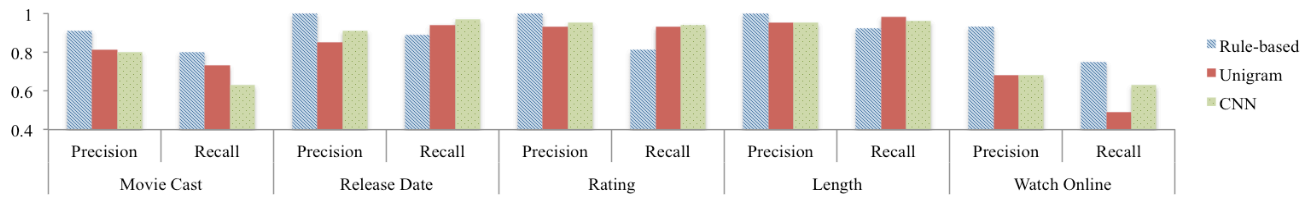

Figure 4: Intent detection results for the low-level intents of `Movie` intent class.

In this set of experiments, we consider the low-level intent classes as query labels. So, the task is a multi-class classification task with 125 classes. It is a challenging classification task, both because there are high number of classes and because some of the low-level intents have similar concepts, e.g. `Person/birthdate` and `Person/age`.

Table 1 summarizes a comparison of different query detection methods. The rule-based system has a mediocre performance although it uses many hand written rules for this specific task. Simple aggregated features of word vectors are even comparable with the rule-based system. The unigram features are good indicators of the query intents; this shows that certain query keywords are good features for certain intents. Even though we did not unify the form of the words (such as stemming or lemmatization)[4], this simple indication of existence of keywords can help to detect query intents. The CNN feature set performs slightly better than the bag-of-word features. Although it is not significantly better, this is an impressive results. It shows that extracted query vectors are meaningful and good representations for queries.

Figures 3 and 4 show precision and recall for a sample set of low-level intents that belong to popular high-level intents of `Person` and `Movie`. The rule-based system is designed for these specific intent classes, so it is expected to perform well. Although the CNN method is based on automatically generated query vectors, it performs similar to the rule-based system on average.

4.3.2 *High-level intent detection*

We repeat the classification experiment with the 14 high-level intent classes this time. For example, the query "barack obama age" will only maps to the `Person` intent. Table 2 shows the results. The trend of results are similar to the low-level intent classification except that the Unigram model achieves the best average precision. In addition, the accuracy of bag-of-word and CNN methods are higher (around 90%), however, their F-scores are lower than the results in Table 1.

4.3.3 *Query clustering*

Finally, we run a case study to see how useful query representation extracted from CNN could be in finding similarity between them. We cluster the queries with their embeddings extracted from the low-level intent classes using k-Means (`k = 125`) algorithm implemented in [13]. Table 5 shows a sample of clusters. It is demonstrated that similar queries are grouped in the same cluster. There are also clusters with the mixture of similar intents. For instance, the last cluster in table 5 contains queries about both movie cast and musical band members. This preliminary experiment reveals some interesting properties about the query vectors. It also provides evidence that the query vectors could be useful for several tasks such as finding emerging query intents by detecting a set of queries that are far from other clusters in the vector space.

[4]For example, the words "age" and "ages" are considered as two different tokens in the unigram feature set.

Table 5: Sample of query clustering using query vectors

Query clusters
george selvie wiki
james brown biography
bob toski bio
ninja turtles youtube
if i stay full movie viooz
watch homeland online
the outlander streaming
john candy death
ryan knight has died
is bruce jenner dead
cast of annie 2014
big bang cast
independence day cast and crew
cast of hollow man
lee rocker band members
santana band members
salif keita band members

5. CONCLUSION AND FUTURE WORK

To understand a user's search query intent, we utilized *query vector representation* which is learned by a convolutional neural network model in a classification task. The major contribution of our work is that it demonstrates that using only query vectors as features is sufficient for query classification, and it simplifies the feature engineering effort. The query vectors give us a meaningful semantic representation of the queries, which helps to detect query intents. They are also helpful to cluster queries. In addition, query vector representation can be used in other tasks related to queries.

In the future, we would like to explore a combination of CNN features with other engineered features like features extracted from query logs. It will also be interesting to use other models to exploit query vectors such as recurrent neural networks with long short-term memory networks (LSTM) cells.

6. REFERENCES

[1] H. Cao, D. H. Hu, D. Shen, D. Jiang, J.-T. Sun, E. Chen, and Q. Yang. Context-aware query classification. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 3–10. ACM, 2009.

[2] J. C. K. Cheung and X. Li. Sequence clustering and labeling for unsupervised query intent discovery. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 383–392. ACM, 2012.

[3] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[4] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

[5] J. Gao, P. Pantel, M. Gamon, X. He, L. Deng, and Y. Shen. Modeling interestingness with deep neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2014.

[6] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[7] J. Hu, G. Wang, F. Lochovsky, J.-t. Sun, and Z. Chen. Understanding user's query intent with wikipedia. In *Proceedings of the 18th international conference on World wide web*, pages 471–480. ACM, 2009.

[8] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014.

[9] Y. Kim. Convolutional neural networks for sentence classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[10] M. Mendoza and J. Zamora. Identifying the intent of a user query using support vector machines. In *String Processing and Information Retrieval*, pages 131–142. Springer, 2009.

[11] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[12] B. Mitra. Exploring session context using distributed representations of queries and reformulations.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[14] F. Radlinski, M. Szummer, and N. Craswell. Inferring query intent from reformulations and clicks. In *Proceedings of the 19th international conference on World wide web*, pages 1171–1172. ACM, 2010.

[15] X. Ren, Y. Wang, X. Yu, J. Yan, Z. Chen, and J. Han. Heterogeneous graph-based intent learning with queries, web pages and wikipedia concepts. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 23–32. ACM, 2014.

[16] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 101–110. ACM, 2014.

[17] W.-t. Yih, X. He, and C. Meek. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014.