Query Segmentation via RNNs Encoder-Decoder Framework

Yiu-Chang Lin
Rakuten Institute of Technology
Boston, Massachusetts - USA 02110
yiuchang.lin@rakuten.com

Giuseppe Di Fabbrizio
Rakuten Institute of Technology
Boston, Massachusetts - USA 02110
difabbrizio@gmail.com

Ankur Datta
Rakuten Institute of Technology
Boston, Massachusetts - USA 02110
ankur.datta@rakuten.com

ABSTRACT

Query segmentation is the task of segmenting a Web query into adjacent phrases, typically keywords that form noun phrases or concepts that are relevant to the search task. In this paper, we describe a research study and some preliminary experiment results for query segmentation via a Recurrent Neural Network encoder-decoder framework on a public benchmark dataset (Webis-QSeC-10). The resulting segmented queries can be used for several downstream tasks such as improving the performance of relevance ranking in search, better understanding of the query intent, and suggesting queries for auto-completion.

KEYWORDS

Query segmentation, Recurrent neural network, Encoder-Decoder

ACM Reference format:
Yiu-Chang Lin, Giuseppe Di Fabbrizio, and Ankur Datta. 2017. Query Segmentation via RNNs Encoder-Decoder Framework. In *Proceedings of SIGIR 2017 eCom, Tokyo, Japan, August 2017,* 5 pages.

1 INTRODUCTION

Query segmentation aims to detect semantically consistent phrases that identify entities and concepts in Web search queries e.g., "**[air conditioner][remote control]**", "**[compact][microwave oven]**", and "**[iphone 7][cover]**". Such phrases are the semantic structural units of a search task and can be exploited by search engines as indivisible units in order to improve retrieval precision or reformulate phrase-level query. It is often the case that short text as in Web queries do not follow grammar rules hence traditional methods based on well-formed English are not applicable.

Query segmentation is one of the most important tasks toward query understanding, a key component of modern search engines for precisely inferring the users' intent through queries since query segments can be further refined into named-entities and semantic relations linking head-phrases with modifiers.

Both supervised and unsupervised learning techniques have been used to solve the query segmentation task in the past. In the supervised learning category, Support Vector Machines ranker [10] was used to learn a structured classifier that makes a segmentation decision (*yes or no*) between each pair of continuous tokens [3]. Another well-known model that has been successfully applied to a variety of sequence labeling task is Conditional Random Fields (CRFs) [14]. CRFs model the conditional probability distribution over a label sequence given an input query where each token in the query is assigned to a label from the possible values of a pre-defined label sets [22]. However, such supervised methods require a huge amount of human segmentation labels which are usually expensive to obtain and, furthermore, careful feature engineering plays an important role in achieving high segmentation accuracy.

On the other hand, in the unsupervised learning family, several methods have been proposed to either automatically collect segmented queries or train segmentation models from query log data. For example, in the e-commerce domain, query terms are aligned to product attribute terms via user's click data and the ambiguities are resolved using frequency and similarity statistics [12]. Statistical methods based on point-wise mutual information (PMI) [13], *n*-gram frequency [18], or Multi-Word Expression probability [16] are also popular. One unsupervised approach using generative language models and Wikipedia as external resource has been reported to have competitive performance [21]. Another unsupervised probabilistic model was proposed to exploit user click-throughs for query segmentation and the model parameters were estimated by efficient expectation—maximization (EM) algorithm [15].

Recently, Deep Neural Networks (DNNs) models have shown its powerful capability to achieve excellent performance on various difficult Natural Language Processing learning tasks. Especially in end-to-end sequence learning tasks, the Encoder-Decoder network [20] that makes minimal assumptions on the sequence structure is widely used in machine translation [1, 4, 5]. In this paper, we propose to treat query segmentation as a machine translation task and apply the Encoder-Decoder framework to generate query segments. Preliminary results on the Webis-QSeC-10 [1] dataset are reported.

2 DATA

The Webis Query Segmentation Corpus (Webis-QSeC-10) [8] consists of 53,437 web queries and each query has at least 10 segmentations provided by 10 different annotators crowdsourced via Amazon's Mechanical Turk (AMT). A sample of 4,850 queries is published as the training set and the remaining 48,587 queries serve as the testing set, with a 1:9 train/test split ratio. The Webis-QSeC-10 is sampled from the subset of the AOL query log [19] which consists of only queries with length from 3 to 10 words. Since 1-word queries cannot be segmented anymore and 2-word queries are typically handled well by proximity features, queries with just 1 or 2 word are excluded. The sampling maintains the query length distribution and the query frequency distribution of the entire AOL query log.

[1] https://www.uni-weimar.de/de/medien/professuren/medieninformatik/webis/corpora/webis-qsec-10/

In: J. Degenhardt, S. Kallumadi, M. de Rijke, L. Si, A. Trotman, Y. Xu (eds.): Proceedings of *the SIGIR 2017 eCom workshop, August 2017, Tokyo, Japan*, published at http://ceur-ws.org

An example query with its segmentations from the training set is shown below, where 1004073900 is the unique query id followed by a list of vote and segmentation pairs indicating the 10 different decisions the AMT workers made for that query.

- `1004073900`
- `(5, 'graffiti fonts|alphabet'),`
  `(3, 'graffiti|fonts|alphabet'),`
  `(2, 'graffiti fonts alphabet')`

Since each query is segmented by at least 10 annotators and not all of them always agree with each other, to select the reference annotation, we apply the *break fusion* strategy described in [7]. The underlying idea is that annotators should at least agree on specific important segments even if there is no absolute majority on the entire query segmentation. Break fusion simply follows the majority of annotators at each single break position of a query. A break is inserted in case of a tie vote. Considering the following example annotation,

- `5 graffiti fonts|alphabet`
- `3 graffiti|fonts|alphabet`
- `2 graffiti fonts alphabet`

at the first break position (between graffiti and fonts), 7 (5+2) annotators agree with no break. Similarly, 8 (5+3) annotators agree with inserting a break at the second break position (between fonts and alphabet). Therefore the final reference is

- `graffiti fonts|alphabet`

3 METHODS

In this section, we describe one baseline method [7] and two models, Conditional Random Fields (CRFs) and Recurrent Neural Networks encoder-decoder framework, which are used in this paper for the query segmentation experiment.

3.1 Wikipedia Titles and Strict Noun Phrases Baseline

This baseline method is simply treating only Wikipedia titles and strict noun phrases as query segments. If the query contains more than one overlapping Wikipedia title, the decision rule proposed in [8] is used, which basically assigns each title a score based on the frequencies in the Google n-gram corpus and multiplied by its length. For strict noun phrases, similarly, the multiplication of their Web frequencies and length is assigned as the score. Finally, the segmentation with the highest score is chosen.

3.2 Conditional Random Fields

Conditional Random Fields have been widely used in NLP structured prediction tasks, especially sequence labeling such as part-of-speech (POS) tagging and named-entity recognition (NER). Formally, let the input sequence $\mathbf{x} = x_1, x_2, ..., x_n$ and label sequence $\mathbf{y} = y_1, y_2, ..., y_n$, we want to model the conditional distribution $P(\mathbf{y}|\mathbf{x})$ so that the optimal label sequence can be predicted by solving $\mathbf{y}^* = \underset{\mathbf{y}}{\text{argmax}} P(\mathbf{y}|\mathbf{x})$. The probabilistic model for sequence CRFs defines a family of conditional probability $P(\mathbf{y}|\mathbf{x}, \boldsymbol{\lambda})$ over all possible label sequences $\mathbf{y}$ given $\mathbf{x}$ with the following form:

$$P(\mathbf{y}|\mathbf{x}, \boldsymbol{\lambda}) = \frac{exp \sum_{i=1}^{n} \sum_j \lambda_j f_j (y_{i-1}, y_i, \mathbf{x}, i)}{Z(\mathbf{x})}$$

$$Z(\mathbf{x}) = \sum_{\mathbf{y} \in Y} \sum_{i=1}^{n} \sum_j \lambda_j f_j (y_{i-1}, y_i, \mathbf{x}, i)$$

where $\boldsymbol{\lambda}$ is the model parameters, f_j is the feature function and the numerator of $P(\mathbf{y}|\mathbf{x}, \boldsymbol{\lambda})$ is composed of potential functions. $\boldsymbol{\lambda}$ can be obtained by maximizing the logarithm of the likelihood of the training data with L_1 or L_2 regularization terms,

$$\mathcal{L}(\lambda) = \sum_i logP(\mathbf{y}|\mathbf{x}, \boldsymbol{\lambda})$$

In order to apply CRFs to the query segmentation task, we introduce the standard *Begin, Inside, Outside* (BIO) tagging schema to maps a segmented query to a sequence of tags. Table 1 shows some example queries from Webis-QSeC-10 training set with their corresponding BIO tags.

segmented query	BIO tagging
graffiti fonts | alphabet	graffiti (B) fonts (I) alphabet (B)
stainless steel | chest freezers	stainless (B) steel (I) chest (B) freezers (I)
rutgers | online | graduate classes	rutgers (B) online (B) graduate (B) classes (I)
review | on | breezes	review (B) on (B) breezes (B)

Table 1: Example queries from Webis-QSeC-10 training set and their corresponding BIO tags.

3.3 Recurrent Neural Networks

The fundamental idea of Recurrent Neural Networks is that the network contains a feed-back connection as shown in the left part of Figure 1, so that it can make use of sequential information. RNNs perform the same task for every element in a sequence $\mathbf{x}$, with the output $\mathbf{o}$ being dependent on the computations from the previous state $\mathbf{s}$. This characteristic enables the networks to do sequence processing and learn sequential structure information. Theoretically, RNNs are capable of capturing arbitrarily long distance dependencies, but in practice, they are limited to looking back only a few steps, known as the gradient vanishing/exploding problem [2].

The right part of Figure 1 shows a typical RNN and its forward computation structure after being unfolded into a full network within the sequence window $t-1$, t, and $t+1$. Assume that the input sequence $\mathbf{x}$ is a sentence consisting of n words, $x_1, x_2, ..., x_n$. x_t is the input token at position t and it can be represented as a typical one-hot vector or a word embedding of dimension d. s_t, the corresponding hidden state or "memory", is calculated based on the previous hidden state and the input at the current step. In this case, we would like to predict the next word given $x_1, x_2, ..., x_{t-1}$ so o_t would be a vector of probabilities across the vocabulary. The following equations explicitly explain the computation of RNNs.

$$s_t = f(Ux_t + Ws_{t-1})$$

$$o_t = softmax(Vs_t)$$

where the function f is a nonlinearity mapping such as tanh or ReLU. U, V and W are matrices (model parameters) and can be

optimized through back propagation. Usually, s_{-1}, which is required to calculate the first hidden state, is initialized to a zero vector.

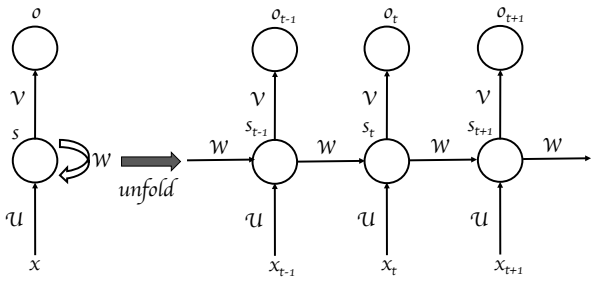

Figure 1: A recurrent neural network and the unfolding in time of the computation involved in its forward computation.

3.4 RNNs Encoder-Decoder Framework

Figure 2 shows a typical encoder decoder framework, a model consisting of two separate RNNs called the encoder and the decoder. The encoder reads an input sequence one item at a time, and outputs a vector at each step (ignored in Figure 2). The final output of the encoder serves as the context vector and the decoder uses this context vector to generate a sequence of outputs. In the context of machine translation, the encoder first processes a variable-length input word sequence from the source language and builds a fixed-length vector representation (context vector). Conditioned on this encoded representation, the decoder produces a variable-length word sequence in the target language. In an ideal case, the context vector can be considered as the meaning of the sequence in latent semantic space, and this idea can be extended beyond sequences. For example, in image captioning tasks, the encoder decoder framework takes the image as input and produces a text description as output. In the reverse direction, image generation tasks take a text description as input and output a generated image.

To fit the query segmentation task into encoder decoder framework, we treat the original query as an input sequence from one language and the segmented query as an output sequence from the other language. The vocabulary size is therefore the same for both languages except that the target language has one additional break token, i.e.,

$$Vocab_{target} = Vocab_{source} + \{"|"\}$$

In practice, the queries and their segmentations combined are treated as a parallel corpus for training. In testing phase, the encoder first calculates the context vector and then generates output tokens one at a time from $Vocab_{target}$.

4 EXPERIMENTS

The Webis-QSeC-10 corpus [8] comprises 53,437 web queries and each of them has at least 10 segmentations. The reference segmentation is obtained as described in *Section 2*. There are 4,850 queries in the training set and 48,587 queries in the testing set. To quantify the segmentation result of different algorithms, we adopt query

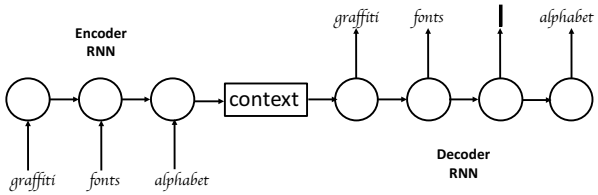

Figure 2: A RNN encoder decoder framework and its application to query segmentation.

level and break level accuracy [7] as the evaluation matrices. At query level, given a query q, its reference segmentation S and the output segmentation S' from the model, the *query accuracy* is 1 if $S' = S$ and 0 otherwise. At break level, a decision whether a break needs to be inserted is made for every two consecutive words in the query. The *break accuracy* is defined as the ratio of correct decisions over all break positions in q with respect to S'. Theoretically, there exists 2^{k-1} valid segmentations for each q, and $\frac{(k^2-k)}{2}$ potential segments that contain at least two keywords from q.

4.1 Model Parameters

In our experiment, we use CRFsuite [2] [17] for optimizing the CRF model parameters and the following set of word uni-gram and bi-gram features are utilized:

- uni-gram: x_{-2}, x_{-1}, x, x_1, $x2$
- bi-gram: $x_{-1}x$, xx_1

For RNN encoder decoder, the following loss function, optimizer and parameters are used:

- Word representation: 1-hot vector
- RNN hidden layer size: 1024
- RNN number of layers: 2
- RNN activation function: tanh
- Loss function: Negative log likelihood loss
- Optimizer: Adam optimizer
- Learning rate: 0.0001
- Dropout rate: 0.05
- Epochs: 50,000

4.2 RNNs Encoder-Decoder Loss

Parameters optimization is obtained by Adam optimizer with negative log likelihood as the loss function. Adam optimizer (Adaptive Moment Estimation) [11] is an algorithm for first-order gradient-based optimization of stochastic objective functions through computing adaptive learning rates for each parameter. Adam keeps an exponentially decaying average of both past gradients and squared gradients. The loss function value on the training set is recorded every 200 epochs and it shows that the training loss decreases steadily with the number of epochs and eventually converges at the end (Figure 3).

[2] http://www.chokkan.org/software/crfsuite/

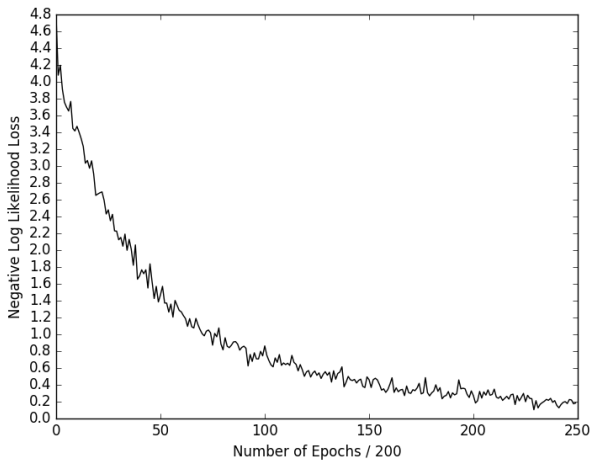

Figure 3: Negative log-likelihood loss for the RNN encoder-decoder on the training set. The loss is recorded every 200 epochs.

4.3 Results

Table 2 shows query-level and break-level accuracy of Wikipedia titles (WT), Wikipedia titles + strict noun phrases (WT+SNP), Conditional Random Fields and RNN encoder-decoder. WT+SNP has the best accuracy among the four methods at both levels. CRF performs better than WT baseline in terms of query level and break level accuracy. The RNN encoder-decoder framework, however, in this case does not perform as expected as it does in other tasks such as machine translation and image captioning.

	query accuracy	break accuracy
WT [7]	0.431	0.769
WT+SNP [7]	0.585	0.837
CRF	0.465	0.814
RNN Encoder-Decoder	0.421	0.664

Table 2: Query level and break level accuracy on Webis-QSeC-10 test set.

5 DISCUSSION

The first two methods (WT and WT+SNP) in Table 2 are unsupervised but require external knowledge resource, e.g., Wikipedia titles, Google n-gram frequencies and Web n-gram frequencies. On the other hand, both CRFs and RNNs encoder-decoder are supervised machine learning methods relying on human annotation. Since the training set only consists of 4,850 annotated queries, which is 1/9 the size of testing set in Webis-QSeC-10, supervised methods cannot benefit from a large amount of training data. In addition to the small size of training set, short-query length is also another key factor that limits the power of RNNs encoder-decoder in query segmentation. Web queries are typically short and less structured compared to standard sentences in machine translation corpus. Therefore, RNNs' remarkable capacity of capturing long-distance dependency is not that effective in this task. Although CRFs outperforms RNNs encoder-decoder, one disadvantage of CRFs is that it requires human-designed features as opposed to RNNs which require no feature engineering.

6 CONCLUSION AND FUTURE WORK

Query segmentation is crucial for a search engine to better understand query intent and return higher quality search results. This paper provides a study on fitting query segmentation task into a RNN encoder-decoder framework and describes preliminary experimental results compared with other baselines. The RNNs does not perform as expected due to the lack of training data and the short nature of query length. However, three feasible future directions might be helpful for improving RNNs encoder decoder framework on query segmentation.

The first direction is to automatically collect a large amount of segmented queries via user implicit feedback from query logs as proposed in [12]. This will solve the challenge of little training data mentioned in *Section 5*. Another direction is to replace the RNN units in the encoder decoder framework with GRUs [6] or LSTMs [9] and add an attention mechanism [1] at the encoder, giving the decoder a way to "pay attention" to different parts of the input while decoding. Finally, substituting pre-trained word embedding for the current one-hot word vector will both reduce the input dimension and provide the network with richer word representation.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Martin Potthast from Bauhaus-Universität Weimar for kindly providing the full Webis Query Segmentation Corpus used for our modeling experiments. The authors would also like to thank the anonymous reviewers for their feedback and helpful advice.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).
[2] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* 5, 2 (1994), 157–166.
[3] Shane Bergsma and Qin Iris Wang. 2007. Learning Noun Phrase Query Segmentation.. In *EMNLP-CoNLL*, Vol. 7. 819–826.
[4] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014).
[5] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).
[6] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).
[7] Matthias Hagen, Martin Potthast, Anna Beyer, and Benno Stein. 2012. Towards optimum query segmentation: in doubt without. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 1015–1024.
[8] Matthias Hagen, Martin Potthast, Benno Stein, and Christof Bräutigam. 2011. Query segmentation revisited. In *Proceedings of the 20th international conference on World wide web*. ACM, 97–106.

[9] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[10] Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 133–142.
[11] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[12] Julia Kiseleva, Qi Guo, Eugene Agichtein, Daniel Billsus, and Wei Chai. 2010. Unsupervised query segmentation using click data: preliminary results. In *Proceedings of the 19th international conference on World wide web*. ACM, 1131–1132.
[13] Giridhar Kumaran and Vitor R Carvalho. 2009. Reducing long queries using query quality predictors. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 564–571.
[14] John Lafferty, Andrew McCallum, Fernando Pereira, and others. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, Vol. 1. 282–289.
[15] Yanen Li, Bo-Jun Paul Hsu, ChengXiang Zhai, and Kuansan Wang. 2011. Unsupervised query segmentation using clickthrough for information retrieval. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 285–294.
[16] Nikita Mishra, Rishiraj Saha Roy, Niloy Ganguly, Srivatsan Laxman, and Monojit Choudhury. 2011. Unsupervised query segmentation using only query logs. In *Proceedings of the 20th international conference companion on World wide web*. ACM, 91–92.
[17] Naoaki Okazaki. 2007. CRFsuite: a fast implementation of Conditional Random Fields (CRFs). (2007). http://www.chokkan.org/software/crfsuite/
[18] Nish Parikh, Prasad Sriram, and Mohammad Al Hasan. 2013. On segmentation of ecommerce queries. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM, 1137–1146.
[19] Greg Pass, Abdur Chowdhury, and Cayley Torgeson. A picture of search.
[20] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.
[21] Bin Tan and Fuchun Peng. 2008. Unsupervised query segmentation using generative language models and wikipedia. In *Proceedings of the 17th international conference on World Wide Web*. ACM, 347–356.
[22] Xiaohui Yu and Huxia Shi. 2009. Query segmentation using conditional random fields. In *Proceedings of the First International Workshop on Keyword Search on Structured Data*. ACM, 21–26.