Multitask Learning for Query Segmentation in Job Search

Bahar Salehi
School of Computing and Information Systems
The University of Melbourne
Melbourne, Australia
salehi.b@unimelb.edu.au

Fei Liu
School of Computing and Information Systems
The University of Melbourne
Melbourne, Australia
fliu3@student.unimelb.edu.au

Timothy Baldwin
School of Computing and Information Systems
The University of Melbourne
Melbourne, Australia
tb@ldwin.net

Wilson Wong
SEEK Ltd.
Melbourne, Australia
wwong@seek.com.au

ABSTRACT

In this paper, we present the first attempt to use multitask learning for query segmentation. We use the semantic category of the words as an auxiliary task and show that segmentation improves when the model is also trained to predict the semantic category of the query terms, outperforming benchmark methods over a novel dataset from a popular job search engine. Our further experiments show that the task of modeling the query term semantics performs better as a standalone task, without adding segmentation as an auxiliary task.

CCS CONCEPTS

• **Information systems → Information retrieval query processing**; • **Computing methodologies → Neural networks**;

KEYWORDS

Query Segmentation, Word Embeddings, Neural Information Retrieval, Multitask Learning, Job Search

ACM Reference Format:
Bahar Salehi, Fei Liu, Timothy Baldwin, and Wilson Wong. 2018. Multitask Learning for Query Segmentation in Job Search. In *2018 ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR '18), September 14–17, 2018, Tianjin, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3234944.3234965

1 INTRODUCTION

Users expect search engines to understand their intent on the basis of short queries without overt linguistic structure. Query understanding is an essential component of modern-day search engines, and improves user satisfaction and search quality [6, 11]. Query segmentation (the task of splitting a query into meaningful terms), and query tagging/classification (the task of semantically categorizing the query terms), are two components of query understanding. These tasks are different from named entity recognition as queries are very short and less structured.

As an example, a user may submit the query *data scientist python bay area*, where she is probably looking for a job with the title of *data scientist*, has the skill of programming in *python*, and wants to work in (the San Francisco) *bay area*. Query segmentation involves splitting the query into *data scientist*, *python* and *bay area*, while query tagging would label *data* and *scientist* as a job title, *python* as a skill, and *bay* and *area* as a location.

Accurate query segmentation and tagging enables better query rewriting, query expansion and query refinement, which in turn improves search quality. In our example, the job search engine would be able to retrieve using just the job title, giving less weight to skill, such that *data scientist* jobs will be shown to the user even if there is no mention of *python*. Moreover, the *bay area* can be expanded to its component cities or surrounding areas if the search engine determines that it is a location name.

In this study, we propose the use of multitask learning (MTL) with query segmentation as the main task and query tagging as the auxiliary task. Our hypothesis is that segmentation implicitly requires knowledge of the meaning of the terms within the query. For example, by knowing that *python* is a skill and *bay area* is a location, our system should be able to conclude with higher probability that these two belong to different query segments. On the other hand, semantic labeling of each term requires implicit knowledge of the tokens in the query. In our example, *data* by itself has a very general skill-related meaning, but in *data scientist*, it forms a job title.

Our contributions in this paper are: (1) we propose the application of MTL for query segmentation in job search; (2) we model query segmentation as a sequential tagging task, with more than one target label sequence; (3) we show that segmentation improves when semantic classification of terms is incorporated as an auxiliary task; and (4) our analysis shows that semantic tagging preforms better when it is considered as a single task. All code from this paper is available at https://github.com/liufly/query-segmentation.

ICTIR '18, September 14–17, 2018, Tianjin, China

ACM ISBN 978-1-4503-5656-5/18/09...$15.00
https://doi.org/10.1145/3234944.3234965

2 LITERATURE REVIEW

2.1 Query Segmentation

Previous studies on query segmentation have mostly been based on mutual information for segment boundary detection [7, 9, 20], or CRF-based sequence classification models [6, 12]. There have also been studies that use handcrafted features, e.g. based on POS tags [3].

[10] is the most similar work to this study, and based primarily on the findings of [15], where word2vec-based word embeddings [18] were shown to implicitly capture the co-occurrence of words, meaning that the components of a given collocation tend to be assigned similar representations. For each pair of words in a query, [10] use a classifier to predict whether there should be a segmentation boundary between them or not, based on the word embeddings of the two words in each pair. Our study is different to that of [10] as they focus on segmentation only, while we target both segmentation and semantic tagging, and they use a large (50K) hand-tagged set of queries, while we use a smaller, noisier training set, automatically sourced from a knowledge based method (explained in Section 4.1).

2.2 Multitask learning

Multi task learning (MTL) has received a lot of attention recently, and has been shown to be useful in many NLP tasks [4, 13, 16, 17], especially when training data is limited [2]. In most MTL approaches, there is one main task with one or more auxiliary tasks, although some models are symmetric and involve main and auxiliary tasks of the same value. Usually in MTL models, the parameters of one or more layers of the neural network are shared between the tasks [5]. The auxiliary tasks are shown to regularize the main task model, with the benefit of not overfitting and generalizing better [1]

In this study, the relatedness between the two tasks of segmentation and tagging motivates us to use a MTL approach. This study is the first attempt to use semantic categories of words to improve query segmentation.

3 METHODOLOGY

In this work, we are interested in the task of sequential tagging, with more than one target label sequence. The input typically comes in the form of sequence tuples: $\mathcal{D} = \{(\mathbf{x}^{(n)}, \mathbf{y}^{1,(n)}, \ldots, \mathbf{y}^{K,(n)})\}_{n=1}^{N}$ where $\mathbf{x}^{(n)} = \{x_1^{(n)}, \ldots, x_T^{(n)}\}$ is the n-th input sequence in $\mathcal{D}$ and its corresponding target label sequence $\mathbf{y}^{k,(n)} = \{y_1^{k,(n)}, \ldots, y_T^{k,(n)}\}$ for the k-th task. For notational convenience, hereinafter we omit the superscript denoting the n-th example.

Motivated by the benefits of MTL and the inter-dependency between the two tasks of query segmentation and tagging, we propose a novel architecture to model the former task with the help of the latter. Specifically, taking inspiration from the recent successes of bi-directional long short-term memory condition random fields (Bi-LSTM-CRFs) in sequential modelling [8, 14], we model these tasks with two task-specific CRFs, both taking input from a shared Bi-LSTM.

More formally, we first convert every element in a sequence $\mathbf{x}$ into its embedding $\boldsymbol{x}_t = \Phi(x_t)$, resulting in a sequence $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$. Here, $\Phi(\cdot)$ is an encoding function, mapping the input x_t into a vector $\boldsymbol{x}_t \in \mathbb{R}^h$.

Next, this sequence of embeddings is taken as input to a forward-pass and backward-pass LSTM to generate a forward-pass representation $\overrightarrow{\boldsymbol{h}}_t$ and forward-pass representation $\overleftarrow{\boldsymbol{h}}_t$, respectively. $\overrightarrow{\boldsymbol{h}}_t$ and $\overleftarrow{\boldsymbol{h}}_t$ are then fed into a CRF layer:

$$s(\mathbf{x}, \mathbf{y}^k) = \sum_{t=0}^{T} \boldsymbol{A}^k_{y_t^k, y_{t+1}^k} + \sum_{t=1}^{T} \boldsymbol{P}^k_{t, y_t^k} \qquad (1)$$

where $\boldsymbol{A}^k \in \mathbb{R}^{|\mathcal{Y}^k| \times |\mathcal{Y}^k|}$ is the CRF transition matrix for the k-th task, $|\mathcal{Y}^k|$ is the size of the k-th label set, and $\boldsymbol{P}^k \in \mathbb{R}^{T \times |\mathcal{Y}^k|}$ is a linearly transformed matrix from the forward and backward representations:

$$(\boldsymbol{P}^k_{t,:})^\top = \overrightarrow{\boldsymbol{W}}^k_{hp} \overrightarrow{\boldsymbol{h}}_t + \overleftarrow{\boldsymbol{W}}^k_{hp} \overleftarrow{\boldsymbol{h}}_t + \boldsymbol{b}^k_p$$

where $\overrightarrow{\boldsymbol{W}}^k_{hp}, \overleftarrow{\boldsymbol{W}}^k_{hp} \in \mathbb{R}^{|\mathcal{Y}^k| \times h}$ with h being the size of $\overrightarrow{\boldsymbol{h}}_t$ and $\overleftarrow{\boldsymbol{h}}_t$. While $\boldsymbol{P}^k_{i,j}$ represents the score of the j-th tag at time i, $\boldsymbol{A}^k_{i,j}$ denotes the transition score from the i-th tag to the j-th tag. The probability of the true sequence can be calculated with the scoring function in Equation (1) by normalising the score of the true sequence with the sum of scores of all possible sequences $\tilde{\mathbf{y}}^k$:

$$p(\mathbf{y}^k|\mathbf{x}) = \frac{\exp(s(\mathbf{x}, \mathbf{y}^k))}{\sum_{\tilde{\mathbf{y}}^k \in \mathbf{Y}^k_{\mathbf{X}}} \exp(s(\mathbf{x}, \tilde{\mathbf{y}}^k))}$$

The model is trained to maximise the probability of the gold label sequence. Given that we are interested in the two tasks of query segmentation and tagging (tasks 1 and 2 respectively), we make use of an α value to manually control the balance between the two in the loss function:

$$\mathcal{L} = (1 - \alpha) \times \log p(\mathbf{y}^1|\mathbf{x}) + \alpha \times \log p(\mathbf{y}^2|\mathbf{x})$$

where $p(\mathbf{y}^k|\mathbf{x})$ is calculated using the forward–backward algorithm. Note that the model is fully end-to-end differentiable and the two tasks are trained jointly using 20% of the data as dev set.

At test time, the model predicts the output sequence with maximum a posteriori probability: $\hat{\mathbf{y}}^k = \arg\max_{\tilde{\mathbf{y}}^k \in \mathbf{Y}^k_{\mathbf{x}}} p(\tilde{\mathbf{y}}^k|\mathbf{x})$. Since we are only modelling bigram interactions, we adopt the Viterbi algorithm for decoding.

4 EXPERIMENTAL RESULTS

4.1 Data

We extracted our experimental data from the query log and job listing of the SEEK commercial job search engine.[1] The words in each query are labeled as one of the 6 categories, which are the most common categories in our data: SKILL (e.g., *python, revit*), JOB TITLE (e.g., *software engineer, cleaner*), LOCATION (e.g., *bay area, Melbourne*), SENIORITY (e.g., *apprentice, entry level*), WORK TYPE (e.g., *night shift, contract*), and COMPANY NAME (e.g., *Microsoft, McDonalds*).

To construct the training and development sets, we randomly sampled 7600 queries from among the most common queries. To

[1] No personally identifiable information was used in our experiments.

	Semantic Category			Segmentation			
Method	Acc	micro F1	macro F1	Acc	Prec	Recall	F1
$\alpha = 0.00$	—	—	—	91.80	82.98	78.39	80.62
$\alpha = 0.33$	77.38	77.50	68.90	**92.46**	**87.50**	**80.90**	**84.07**
$\alpha = 0.50$	79.67	79.67	62.80	91.80	85.64	77.89	81.58
$\alpha = 0.66$	76.72	76.72	65.59	91.80	85.08	77.39	81.05
$\alpha = 1.00$	79.67	79.67	64.23	—	—	—	—
Rule-based	48.52	64.07	57.31	82.95	58.13	73.71	65.00
CRFsuite	—	—	—	87.21	75.62	60.80	67.41
[10]	—	—	—	91.80	83.60	79.40	81.44
IOBtagger	78.69	78.69	67.21	90.49	86.34	79.40	82.72

Table 1: Test results on all data

		Semantic Category			Segmentation			
Frequency	α	Acc	micro F1	macro F1	Acc	Prec	Recall	F1
Very low	0.00	—	—	—	88.61	78.26	70.59	74.23
	0.33	68.35	68.35	43.67	89.87	81.40	68.63	74.47
Low	0.00	—	—	—	89.58	76.67	82.14	79.31
	0.33	72.91	72.91	42.81	91.67	78.57	78.57	78.57
High	0.00	—	—	—	89.87	85.71	75.00	80.00
	0.33	69.62	69.62	59.36	89.87	85.71	75.00	80.00
Very high	0.00	—	—	—	91.21	83.02	74.58	78.57
	0.33	82.41	82.41	58.47	95.60	92.73	86.44	89.47

Table 2: Test results on queries with varying frequency

automatically label this data, we used a pre-existing API to a job-specific knowledge base, which uses simple longest string match relative to the knowledge base content to label each query token, subject to the following priority: seniority, work type, skill, job title, company, location. For example, in the case of *chicago university*, *chicago* would be labeled as company rather than location, and in the case of *java developer*, *java* is labeled as skill, despite *java developer* being a job title, because skill has a higher priority.

Any knowledge-based method suffers from a recall problem: with rapid changes in the job market, especially in the emergence of new companies, job titles and skills, the knowledge base inevitably becomes stale. To counter this issue, we removed all tokens from the training queries which are not contained in the knowledge base. Despite the noisy and incomplete data, we use it as training data for our multi-task learning approach, which is known to be effective when training data is scarce.

Our test data set of 200 queries was randomly sampled, stratified based on frequency in the query log.[2] The queries were first labeled using the knowledge base API, and then manually checked and post-corrected. From the 200 selected queries, we chose to discard 40 because they contained a term that did not belong to any of the 6 common semantic categories which are the focus of this study (such as person's name, phone number, job id, or very general terms such as *water*). In the case of misspellings, we labeled as if the correct spelling were used. In the case of *cientist*, e.g., we would label it as if it were *scientist*.

4.2 Baselines

To compare our proposed method with previous research, we use four baselines. Our first baseline is the rule-based method use to automatically label the training data. The second baseline is based on the work of [10], where the concatenated embeddings of each adjacent token bigram in the query are fed into a logistic regression classifier to label segmentation boundaries. We select CRFsuite [19] as our third baseline, taking word embeddings as feature input. In our fourth baseline (IOBtagger), we use BiLSTM+CRF as a single-task classifier, combining the segmentation and class category labels using IOB tags, akin to chunking or named entity recognition.

Word embeddings were trained over a corpus of job ads, containing a total of 422 million tokens. We used the CBOW variant of word2vec [18], to generate 300-dimensional embeddings, with all other hyperparameters set to the default.

4.3 Evaluation

The results of our experiment are shown in Table 1. CRFsuite, [10] and our method with $\alpha = 0$ are segmentation-only methods, and therefore there are no results for semantic category prediction. On the other hand, when $\alpha = 1$, our proposed method becomes a

[2] 50 random samples selected from each quartile

single task model which predicts the semantic category only, and therefore there are no results for segmentation.

The α parameter in Equation 3 is used to control the contribution of each task in training the model. When $\alpha = 0$, the tagging labels are not used in training. On the other hand, when $\alpha = 1$, the segmentation labels are not used in training the model. As shown in Table 1, when $\alpha = 0.33$, i.e., the weight for segmentation is twice the weight that for tagging, the highest segmentation accuracy is observed. In this case, the segmentation results improve across all metrics compared to when segmentation is considered as a single task ($\alpha = 0$). This confirms our hypothesis that segmentation can improve if the model has some understanding of the words' semantic categories. However, the best accuracy for tagging is observed when it is considered as a single task, showing that segmentation information confuses the semantic tagging model.

Looking at the baselines, the knowledge-based method performs the worst in terms of accuracy, precision, recall, and F-score among all methods (except for CRFsuite's recall). CRFsuite performs considerably better and IOBtagger performs better again, yet our proposed method outperforms both methods across all evaluation metrics for the segmentation task. The second of these results is particularly significant, in that we use the exact same BiLSTM+CRF with the same embeddings, and yet with MTL, the results are substantially higher. However, IOBtagger performs worse than single task model ($\alpha = 1$) in predicting semantic category in terms of accuracy but performs better in terms of macro F-score. Finally, compared with [10], our method is superior only when $\alpha = 0.33$, confirming our hypothesis that the inclusion of semantic category signal in a multi-task learning architecture results in better segmentation performance.

Looking to the results over queries of different frequency in Table 2, we observe the same trend that MTL ($\alpha = 0.33$) performs better than single-task learning ($\alpha = 0$), and achieves higher results for very high frequency queries, but lower segmentation performance for very low frequency queries.

5 CONCLUSIONS

In this paper, we presented the first application of multitask learning to query segmentation and showed that segmentation improves when the model is also trained on the semantic categories of words. Our further analysis shows that unlike segmentation, semantic tagging performs better when the model is trained as a single task.

REFERENCES

[1] Jonathan Baxter. 2000. A model of inductive bias learning. *Journal of Artificial Intelligence Research (JAIR)* 12, 149-198 (2000), 3.

[2] Adrian Benton, Margaret Mitchell, and Dirk Hovy. 2017. Multitask Learning for Mental Health Conditions with Limited Social Media Data. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers.* Valencia, Spain, 152–162.

[3] Shane Bergsma and Qin Iris Wang. 2007. Learning noun phrase query segmentation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL).*

[4] Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers.* Valencia, Spain, 164–169.

[5] Rich Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias1. In *Proceedings of the Tenth International Conference on Machine Learning.* 41–48.

[6] Jiafeng Guo, Gu Xu, Hang Li, and Xueqi Cheng. 2008. A unified and discriminative model for query refinement. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval.* ACM, 379–386.

[7] Jian Huang, Jianfeng Gao, Jiangbo Miao, Xiaolong Li, Kuansan Wang, Fritz Behr, and C Lee Giles. 2010. Exploring web scale language models for search query processing. In *Proceedings of the 19th international conference on World wide web.* ACM, 451–460.

[8] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991* (2015).

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

[10] Ajinkya Kale, Thrivikrama Taula, Sanjika Hewavitharana, and Amit Srivastava. 2017. Towards Semantic Query Segmentation. In *Proceedings of the SIGIR 2017 Workshop on Neural Information Retrieval (Neu-IR'17).*

[11] In-Ho Kang and GilChang Kim. 2003. Query Type Classification for Web Document Retrieval. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval.* 64–71.

[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] Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. 2016. Improving sentence compression by learning to predict gaze. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* 1528–1533.

[14] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2016).* San Diego, USA, 260–270.

[15] Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics* 3 (2015), 211–225.

[16] Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114* (2015).

[17] Héctor Martínez Alonso and Barbara Plank. 2017. When is multitask learning effective? Semantic sequence prediction under varying data conditions. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers.* Valencia, Spain, 44–53.

[18] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proc. ICLR Wrkshp.*

[19] Naoaki Okazaki. 2007. CRFsuite: a fast implementation of Conditional Random Fields (CRFs). http://www.chokkan.org/software/crfsuite/

[20] Knut Magne Risvik, Tomasz Mikolajewski, and Peter Boros. 2003. Query Segmentation for Web Search.. In *WWW (Posters).*