A Topic Model Scoring Approach for Personalized QA Systems

Hamidreza Chinaei¹, Luc Lamontagne¹, François Laviolette¹, and Richard Khoury²

¹ Department of Computer Science and Software Engineering, Université Laval, Québec, Canada
² Department of Software Engineering, Lakehead University, Thunder Bay, Canada

Abstract. To support the personalization of Question Answering (QA) systems, we propose a new probabilistic scoring approach based on the topics of the question and candidate answers. First, a set of topics of interest to the user is learned based on a topic modeling approach such as Latent Dirichlet Allocation. Then, the similarity of questions asked by the user to the candidate answers, returned by the search engine, is estimated by calculating the probability of the candidate answer given the question. This similarity is used to re-rank the answers returned by the search engine. Our preliminary experiments show that the reranking highly increases the performance of the QA system estimated based on accuracy and MRR (mean reciprocal rank).

Keywords: Personalized QA, User Modeling, Topic Modeling.

1 Introduction

In QA systems, the system automatically finds answers to questions asked by the user. For instance, the user may ask How old is the oldest complete genome? The question is processed linguistically and search phrases or keywords are extracted, which are then used to retrieve documents, snippets (the passages returned by the search engine), or sentences. Then a short answer is constructed from the search results and returned to the users. For instance, for the above question, the answer would be 700,000 years old.

In personalized QA systems, the user may ask several questions either to complete a task or to learn several facts about a topic of interest. The user may need to find as much information as possible say about the oldest complete genome. Each time, the system returns an answer and may give extra information or suggestions for other questions of potential interest to the user.

Through its interaction with the user, the personalized QA system can learn incrementally user models based on the user browsed contents and from questions submitted to the system. We assume that a personalized QA system supports some user tasks. While accomplishing their tasks, the users should consult some documents (e.g., news articles). The learned models are then reused in the topic model function that we propose in this paper to estimate the probability of a candidate answer given a question. In this way, the system results can be assessed and prioritized using the learned user model.

Schlaefer introduced the Ephyra question answering engine, a modular and extensible framework that allows to integrate multiple approaches to question answering.
in one system. We use an open version of the Ephyra engine, i.e., openEphyra[^1], and build our personalized QA system on top of it. Our QA system includes for instance a logger that collects the documents read by user, a topic modeler that models the topics of interest to the user, etc.

In particular, we propose a reranking function to increase the evaluation of the passages, returned by the search engine, that may contain the answer. This is because the QA systems have a bigger chance to extract the correct answers if the top retrieved page/nippets contain the answers [6].

To this end, we propose a new probabilistic scoring approach based on the topics of the questions and the candidate answers. First, the set of interesting topics to the user is learned by applying a topic modeling approach, such as Latent Dirichlet Allocation (LDA) [1], on passages of interest to the user. Then, the similarity between questions asked by the user and candidate answers returned by the system is estimated by calculating the probability of the answer given the question. The probabilities are then estimated using the models learned by LDA. This topic model function is added to the set of QA functions (so called filters) that contribute to extracting and ranking the candidate answers to the question.

In the rest of this paper, we describe topic modeling. A brief introduction to the LDA method is presented in Section 2. We then explain in Section 4 our proposed method of scoring the candidate answers based on the topics of the question and those of the candidate answers. In Section 5 we briefly describe the architecture of our personalized QA system. We then explain in Section 6 our experiments and results. Finally, we conclude this work in Section 7.

## 2 Topic Modeling

Topic modeling techniques learn a set of possible topics from text documents of a corpus. The Latent Dirichlet Allocation (LDA) model is the pioneer topic modeling approach with interesting results. LDA is a latent Bayesian topic model which is used for discovering the hidden topics of documents [1]. In this model, a document can be represented as a mixture of the hidden topics, where each hidden topic is represented by a distribution over words occurred in the document.

Suppose we have the sentences shown in Table 1. Using the LDA method, we can automatically discovers the topics that are contained in the given collection (data). For example, given $n$ asked topics, the LDA model learns the topics and the assignment of each topic to each sentence in the given data. The learned topics are represented using the words and their probabilities of occurring within each topic. The topics are presented in Table 2 with their top-10 words. The topic representation for topic A illustrates that this topic is about horse, dna, genome, etc. The topic representation for topic B illustrates that this topic B is about film, festivals, stars, etc. For any given snippet, then the algorithm produces in output a probability distribution over the set of all topics. This distribution is interpreted as the probability that the snippet belong to each of the learned topic, as presented in Table 3. Note that the examples shown in this section are from our collection as well as the results of the experiments of this work.

[^1]: https://mu.lti.cs.cmu.edu/trac/Ephyra/wiki/OpenEphyra