PROBABILISTIC MODELING IN COMMUNITY-BASED QUESTION ANSWERING SERVICES

by

Zeinab Zolaktaf Zadeh

Submitted in partial fulfillment of the
requirements for the degree of
Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
February 2012

DALHOUSIE UNIVERSITY

FACULTY OF COMPUTER SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "PROBABILISTIC MODELING IN COMMUNITY-BASED QUESTION ANSWERING SERVICES" by Zeinab Zolaktaf Zadeh in partial fulfillment of the requirements for the degree of Master of Computer Science.

Dated: February 29, 2012

Supervisor: _____

Readers: _____

_____

DALHOUSIE UNIVERSITY

DATE: February 29, 2012

AUTHOR: Zeinab Zolaktaf Zadeh

TITLE: PROBABILISTIC MODELING IN COMMUNITY-BASED QUESTION ANSWERING SERVICES

DEPARTMENT OR SCHOOL: Faculty of Computer Science

DEGREE: M.C.Sc. CONVOCATION: May YEAR: 2012

Signature of Author

To my family.

Table of Contents

List of Tables

List of Figures

Abstract

Community-based Question Answering (CQA) services enable members to ask questions and have them answered by the community. These services have the potential of rapidly creating large archives of questions and answers. However, their information is rarely exploited.

This thesis presents a new statistical topic model for modeling Question-Answering archives. The model explicitly captures topic dependency and correlation between questions and answers, and models differences in their vocabulary.

The proposed model is applied for the task of Question Answering and its performance is evaluated using a dataset extracted from the programming website Stack Overflow. Experimental results show that it achieves improved performance in retrieving the correct answer for a query question compared to the LDA model. The model has also been applied for Automatic Tagging and comparisons with LDA show that the new model achieves better clustering performance for larger numbers of topics.

List of Abbreviations Used

CQA	Community-based question answering
DCoDQ	Common Words over Distinct Query Words
JS	Jensen-Shannon
KL	Kullback-Leibler
LCoLQ	Length of Common Words over Length of Query Words
LDA	Latent Dirichlet Allocation
LSI	Latent Semantic Indexing
MAP	Mean Average of Precision
MRR	Mean Reciprocal Rank
PLSI	Probabilistic Latent Semantic Indexing
QATM	Question Answering Topic Model
SWB	Special Words with Background
TFIDF	Term Frequency Inverse Document Frequency

Acknowledgements

I would like to thank my adviser, Dr Evangelos Milios, whose knowledge, support and encouragement allowed me to develop and pursue this thesis.

I would also like to express my gratitude to Dr Mahdi Shafiei, for his guidance, help and support.

Last but not least, I would like to thank my family, who have loved and supported me throughout my life.

Chapter 1

Introduction

Community-based question answering (CQA) services [1] such as Linux Questions [1], Yahoo! Answers [2], and Stack Overflow [3] have recently become very popular. They enable members to ask questions and have them answered by the community. They provide an alternative to traditional web search, and allow users to directly acquire their information needs from other users. These services have the potential of rapidly creating large archives of questions and answers. A considerable portion of their archive can potentially be used as a valuable resource for the information needs of other people. However, one of the main drawbacks of existing CQA services is that the archive information is rarely exploited [5, 16, 17]. The high presence of redundant questions and answers is an indication. This redundancy may be considered as a failure of users to do proper diligence before asking questions or it might be regarded as a failure of internal (or external) search features in locating information and making it accessible.

Regarding the former, on the one hand, it is often difficult to formulate a question using the terminology appropriate for a particular subject, especially if the asker is unfamiliar with that subject. On the other hand, identifying the right terminology to search the archive can also be problematic. As a result, many semantically similar questions are generated. To avoid duplication and save time and effort for users, an effective search mechanism that can identify the semantics of questions and is capable of locating relevant information is needed.

The main problem with current search features arises from the characteristic of natural language in which semantically similar content can have different literal representations. Applying document representation techniques that rely on word occurrence will generate different representations for such content. Traditional lexical

[1] `http://www.linuxquestions.org/questions/`
[2] `http://answers.yahoo.com/`
[3] `http://stackoverflow.com/`

similarity measures are adequate if sufficient word overlap exists. However, questions and answers on CQAs are typically of a short length and have sparse representations often with little word overlap. The problem is further exacerbated considering the fact that different terminologies are used by users because their knowledge and expertise levels differ. Methods that bridge this vocabulary gap and enhance the representation of the documents by encoding information about their semantic structure are needed.

Moreover, an additional problem that exists on many legacy question-answering archives is that they lack the semantic information needed to browse their content. For such websites, a system that automatically annotates their content with meta data that describe them, would be valuable and could help understand and browse their content.

In this work, we propose a probabilistic topic model for the content of Question-Answering archives. We use the model for the task of Question Answering, in which existing question-answer pairs in the archive are automatically retrieved and ranked given a newly submitted question. The model is also applied for the task of automatic tagging of questions. In the following, we present our model and report experimental results.

Chapter 2

Background and Related Work

One of the main objectives of current research in CQA is to develop methods that automate tasks and help save time and effort of the users, whether they are asking or answering questions. This research includes various areas such as question answering, expertise modeling, automatic tagging, content quality prediction, spam detection, and summarization of content.

CQA services offer two types of information sources that can be used to mine and exploit their content (1) the textual content, such as the content of question and answers (2) the structural or social features, such as votes, click counts, and tags. Structural features are forum-dependent. Appendix B gives a brief summary of these features. Usually textual features, such as content, are used to evaluate the relevance of a document to a query, and non-textual features, such as user votes or user ranks, are used to measure the quality of documents [16].

2.1 Question Answering

2.1.1 Combination of Textual Content and Non-Textual Features

In the question answering task, the archive is examined to locate relevant questions and answers for a newly submitted question. Non-textual features are utilized to predict the quality of documents in [16]. This quality measure is then incorporated into the language modeling-based retrieval model to retrieve high quality as well as relevant question-answer pairs. Quality ranking is cast as a binary classification problem and stochastic gradient boosted trees are used to classify the content into high-quality and low-quality content in [1]. Features are constructed using textual elements, usage data (clicks), and community ratings. A ranking framework that combines both relevance and quality is presented in [5], where the regression-based gradient boosting framework is applied for the problem of learning a ranking function

for question-answer pair retrieval. A combination of social features (i.e. user interaction activities), textual features (i.e. textual similarity and overlap between query, question and answers), and statistical features (i.e. independent features like the number of tokens for query, question and answers) is used to represent each query-question-answer triplet. To train the ranking function, votes obtained from users are used to extract preference data. The dataset consists of a query set extracted from TREC, where, for each query, relevant question-answer pairs are extracted from Yahoo! Answers. The approach is later modified to make it more resilient to vote spam [4].

The aforementioned approaches rely on community-based elements (e.g. user votes, user interaction information) which may or may not be offered by some CQA services. In contrast, a variety of techniques focus on the abundant textual information offered by the forums.

2.1.2 Textual Content

A variety of techniques utilize the textual content available on CQA services to address problems such as question answering. For these approaches, an important issue that must be resolved is how to make information in the text collection accessible. Text can contain information at different levels of granularity, from simple word-based representations, to rich hierarchical syntactic representations, to high-level semantic representations obtained using document collections. The right level of analysis must be chosen with respect to the properties of the text collection.

The vector space or bag of words model [25, 26] is a word-based representation model. In this model, each document is viewed as a vector of words where weights for the words are assigned based on term occurrence statistics. TFIDF term weighting is one of the most common schemes for computing these weights. A problem associated with this model is the generation of high-dimensional sparse vectors in which terms are assumed to be independent. Dimension reduction techniques such as Latent Semantic Indexing (LSI) [10] and Probabilistic Latent Semantic Indexing (PLSI) [15] aim to map the high dimensional vector representation into a lower dimension latent semantic space. To compare documents, a matching function such as cosine similarity is generally used.

Language models [24] also analyze documents on a word-based level. However, they integrate document indexing and document retrieval into one model. For each document in the collection they infer a probability distribution referred to as the language model of the document. To retrieve similar documents to a query, they estimate the probability of generating the query according to the language model for each document.

There are two assumptions in both these word-based models that may limit their application for CQA services: 1) The main intuition in both approaches is that queries are composed of distinguishing keywords that allow the model to retrieve relevant documents. For community question answering services, this notion implies users know what they are looking for and can choose query terms that distinguish documents of interest from the rest of the collection. However, because of differing levels of knowledge and technical competency, users may or may not generate such queries 2) In addition, sufficient overlap between the document and query vectors is required to derive similarities. But a significant property of the content on CQA's is that they are short with many linguistic variations.

To overcome the word-mismatch problem, documents can be represented using a finer level of granularity. To find similar questions, a retrieval framework based on syntactic tree structure is proposed in [29]. They use structural representations of questions to encode lexical, syntactic, and semantic features into the matching model.

An alternative approach is to augment document representations with semantic information. This information is either extracted from an external resource or derived from text automatically. A question answering system that combines statistical similarities with semantic information for retrieving existing answers in frequently asked questions (FAQs) is described in [7]. Conventional vector space models are used to calculate the statistical similarity and WordNet [11] is used to estimate the semantic similarity. The question answering process is broken down to sub-procedures that include question analysis, document retrieval, answer extraction and answer selection in [18]. A probabilistic answer selection framework that uses external semantic sources is proposed. In particular, logistic regression is used to estimate the probability that an answer candidate is correct given features obtained from external resources.

There are also statistical techniques that use word patterns to derive semantic

information from the text collection itself. This information is then integrated into the retrieval framework. A statistical machine translation model for retrieving questions from a CQA archive that are semantically similar to a user's question, is proposed in [17]. In this approach, the lexical mismatch problem between questions is overcome by implicitly expanding queries with the help of translation models. In particular, translation probabilities are integrated into a query likelihood model.

2.1.3 Topic Models

Probabilistic topic models [6, 13] can derive semantic information from text automatically, on the basis of the observed word patterns. These unsupervised probabilistic models use the co-occurrence structure of terms in text collections to recover the latent topic structure and map items of different literal representation into a smaller topic space. This topical representation of text allows modeling of linguistic phenomena like synonymy and polysemy and lessens the semantic relatedness problems inherent in natural language.

Latent Dirichlet Allocation (LDA) [6] is a widely used topic model. It treats words in a document as multinomial observations. For each document, a distribution over topics is estimated where each topic is a distribution over the words in the corpus. The derived topics allow the model to map the sparse high dimensional literal representations into a lower dimensional topic space. The similarity between documents and queries can then be computed at a semantic level using the distance between documents and queries in this topical space.

The use of topic models for information retrieval tasks is investigated in [30]. Experimental results show that directly employing the output of topic models hurts retrieval performance. The reason for this is over-generalization; topic models provide coarse-grained topics that have been inferred using the whole corpus. Consequently, an ad-hoc method that combines LDA and query-likelihood retrieval is presented. The utility of different types of topic models for information retrieval is also explored in [31]. The research examines the practicality of sophisticated topic models for document retrieval. An important conclusion of this work is that topic models are helpful when query topics are very broad, even when different topic models are combined to represent fine-grained topics.

A different approach is proposed in [9] and a topic model that can handle the trade-off between general and specific information in documents is proposed. The model is called special words with background (SWB) model. A novel similarity measure that utilizes semantic-level similarities based on hidden concepts on top of lexico-syntactic features is described in [8].

The idea of using the thematic structure of a document collection as indications of the semantics of the collection has also been employed for the task of expertise modeling. Capturing the expertise of users has a number of applications, most importantly it allows questions to be automatically directed to experts who can ask answer them. A probabilistic framework for predicting the best answers for new questions is proposed in [20]. The answering history of answerers is tracked and their interests are modeled using a combination of Language Modeling and LDA. The activity and authority of users is also considered to find active users who can give accurate answers.

The Author-Topic-Persona model is introduced in [23]. It addresses the task of matching reviewers with scientific papers. The expertise of reviewers is modeled based on papers they have written in the past. Each author is assumed to have one or more personas, which are represented as independent distributions over hidden topics. For a new paper, they rank the reviewers based on the likelihood of the words of the new paper under each reviewer's distribution.

2.2 Automatic Tagging

Many resources and websites allow users to collaboratively tag their content. Tags categorize similar content and facilitate browsing and search. However, most resources with this feature often lack a predefined tagging vocabulary. As a result, a variety of different tags are generated by users. Three major problems are associated with current tagging systems [12]: polysemy, synonymy and level variation. Polysemy refers to the case where a single tag can have multiple meanings. Synonymy refers to instances where multiple tags with the same meaning are used. The third problem is the inconsistency of tags in terms of specificity or generality; users do not use the same level of abstraction. In addition, many unique but rare tags are generated by users; tags that are inefficient for browsing or searching the archives. One of the main reasons behind these problems is that users do not have any indications regarding the

topical structure of the archive.

Hence, a mechanism that provides insights regarding the archive's topical structure, or the tagging vocabulary in use, can help alleviate the aforementioned problems and save time and effort of the users. The problem of automatically grouping questions on Yahoo! Answers into predefined categories is addressed in [22]. The effectiveness of K-means and PLSI for solving this problem is investigated. The work concludes that incorporating user information, such as areas of expertise , can improve the outcome of clustering. In a different approach [14], LDA is applied to a corpus of about 20000 news stories and a clustering of the corpus is obtained. The resulting document-topic associations are then compared to categories assigned manually to them.

Chapter 3

Methodology

3.1 Question Answering Topic Model

Word mismatch refers to the phenomenon in which a concept is described by different terms in user queries and in source documents. Questions and answers on CQAs are typically short in length and have sparse representations often with little word overlap. To resolve the word mismatch problem in this domain, one approach is to bridge the vocabulary gap by encoding information about the semantics of documents.

To enhance the representation of questions and answers, and encode information about their semantics we propose a new topic model. Our model builds upon the common assumption in topic models [6] that a document is a mixture of topics, where each topic is defined to be a distribution over words. This assumption is appropriate for data from CQA services because questions are typically assigned multiple tags or topics. Furthermore, it is natural to expect that topics in the answers are influenced by topics in the question. However, subjects raised in answers are typically more technical and specific. This is because the knowledge and expertise of the answerers and askers differs; answerers, who can be regarded as experts on the subjects, are more likely to use terms appropriate for the particular realm of knowledge whereas the askers may use less technical terminology. Answers may also contain additional topics that are correlated to the topics in the question, topics that the asker was unaware of and are not explicitly contained in the question. For instance, given a question about string manipulation, the answer might contain topics such as regular expressions or pattern matching. Additional features relevant to text processing languages such as Python or Perl may also be introduced by the answerer (Figure 3.1).

A simple topic model such as LDA [6] is incapable of modeling the dependencies between topics in the answers and topics in questions and may therefore prove to be ineffective for such a setting. The aforementioned aspects of topics in question and answers, emphasize the need for a model that distinguishes between topics in

Question:
`How do I replace all occurrences of a word in a document with another word? Any solution is welcome!`
Answer:
`Regular expressions(regex) allow you to search and manipulate text based on patterns. In some languages, standard string manipulation functions are available e.g. replaceAll() method from Java's string class. In other languages, such as Perl, regex's are integrated into the language itself. Utilities such as grep, vi can also perform this type of pattern matching and replacement.`

Figure 3.1: Questions and answers on CQA's exhibit multiple topics. Topics in answers are influenced by topics in questions.

questions and answers and that can capture topic dependency and correlation across the whole corpus.

Using this intuition we introduce a model that incorporates two types of latent variables, question topic (Q-topics) and answer topic (A-topics). We refer to our model as Question Answering Topic Model or QATM. The two types of topics allow us to model the differences in the vocabulary of questions and answers. They also allow the model to capture the correlation between topics.

Q-topics (β_Q) and A-topics (β_A) are Multinomial distributions over distinct vocabularies for questions and answers respectively. We assume that there are K Q-topics and L A-topics. Each word ($W^n_{Q_i}$) in question Q_i is assigned to a Q-topic $Z^n_{Q_i}$ drawn from a Multinomial distribution θ_{Q_i} over Q-topics.

Each word ($W^n_{A_{i,j}}$) in answer j of question i is assigned to an A-topic ($Z^n_{A_{i,j}}$) that is conditioned on a Q-topic ($Y^n_{A_{i,j}}$). This Q-topic is drawn from the topic distribution of the corresponding question. By conditioning A-topics in an answer on Q-topics drawn from the topic distribution of the corresponding question, topics in answers are influenced by topics in the question and the model captures such a dependency. This is done through the latent variable ϕ, a $K \times L$ matrix. Each row k in ϕ defines mixture weights for A-topics corresponding to Q-topic k. This results in each Q-topic being associated with a distribution over A-topics. Dirichlet priors are defined over all θ_{Q_i} and rows in β_Q, β_A and ϕ with parameters α_θ, α_{β_Q}, α_{β_A} and α_ϕ respectively. We use the plate notation to show the QATM model in Figure 3.2.

The generative process for the model, which explains how observations could have been generated by realizations of random variables and their propagation along the

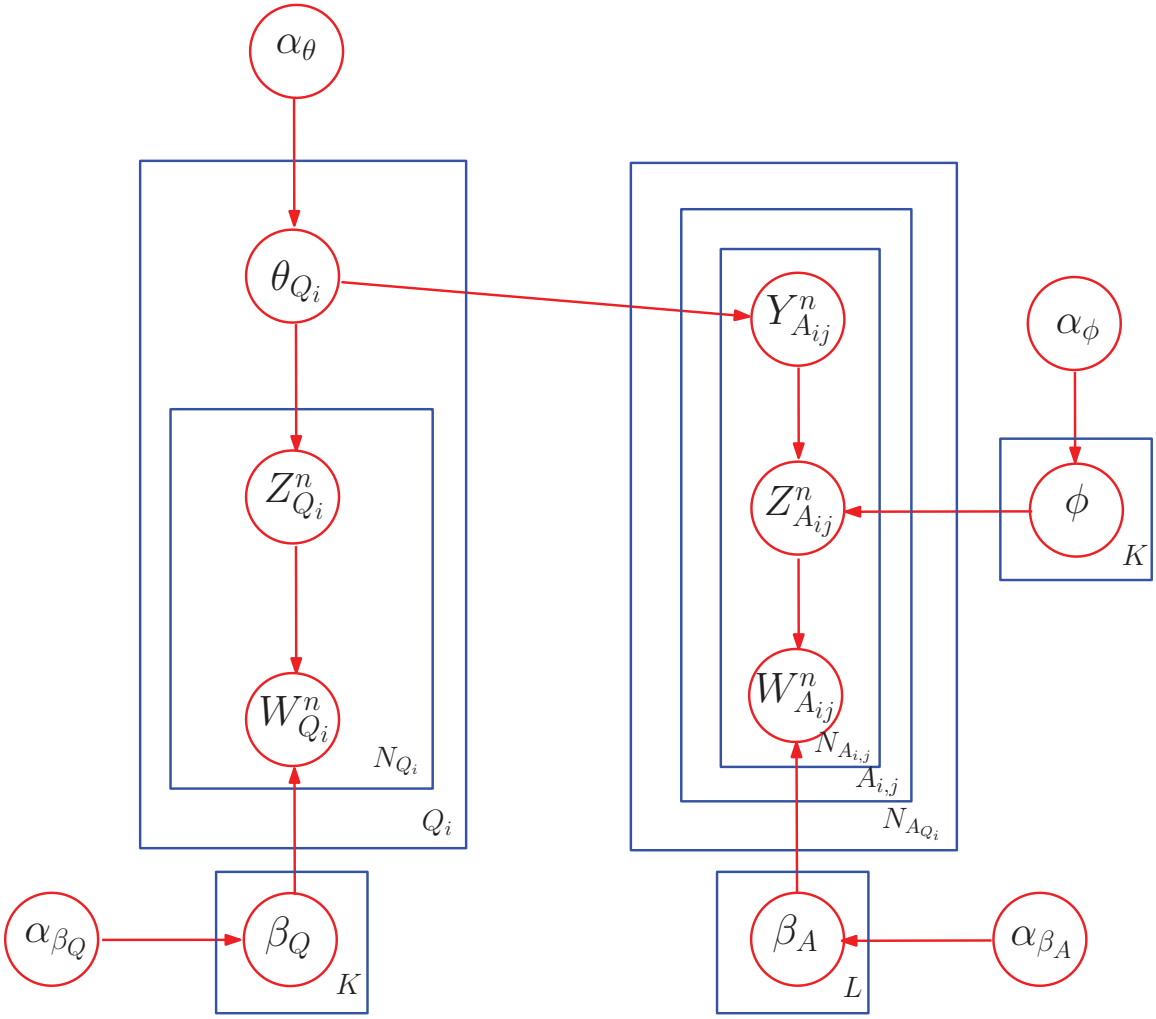


Figure 3.2: Question Answering Topic Model (QATM). The model explicitly captures topic dependency and correlation between questions and answers, and models differences in their vocabulary.

directed edges of the network, is given in Algorithm 1 while Figure 3.3 gives a list of all involved quantities. The plate diagram in Figure 3.2 represents the joint distribution of all known and hidden variables given the hyperparameters, for a single document:

$$\begin{aligned}
&\mathrm{P}(W_{Q_i}, W_{A_i}, Z_{Q_i}, Y_{A_i}, Z_{A_i}, \theta_{Q_i}, \phi, \beta_Q, \beta_A | \alpha_\theta, \alpha_{\beta_Q}, \alpha_{\beta_A}, \alpha_\phi) = \\
&\left(\mathrm{P}(\theta_{Q_i}|\alpha_\theta) \left(\prod_{n=1}^{N_{Q_i}} \mathrm{P}(Z_{Q_i}^n|\theta_{Q_i}) \mathrm{P}(W_{Q_i}^n|Z_{Q_i}^n, \beta_Q) \right) \right) \\
&\prod_{j=1}^{N_{A_{Q_i}}} \prod_{n=1}^{N_{A_{i,j}}} \mathrm{P}(Y_{A_{i,j}}^n|\theta_{Q_i}) \mathrm{P}(Z_A^n|Y_{A_{i,j}}^n, \phi) \mathrm{P}(W_{A_{i,j}}^n|Z_{A_{i,j}}^n, \beta_A) \\
&\prod_{k=1}^{K} \mathrm{P}(\beta_Q^k|\alpha_{\beta_Q}) \prod_{l=1}^{L} \mathrm{P}(\beta_A^l|\alpha_{\beta_A}) \prod_{k=1}^{K} \mathrm{P}(\phi_k|\alpha_\phi)
\end{aligned} \tag{3.1}$$

where Z_{Q_i}, Y_{A_i}, Z_{A_i}, θ_{Q_i}, ϕ, β_Q, and β_A are the hidden variables. The input parameters include $\alpha_\theta, \alpha_{\beta_Q}, \alpha_{\beta_A}, \alpha_\phi$. W_{Q_i} and W_{A_i} are the observations which are given for a single question and its answers. To obtain the joint distribution of all known and hidden variables given the hyperparameters, for the entire corpus, Equation 3.1 can be rewritten as a product over all documents in the entire corpus:

$$\begin{aligned}
&\mathrm{P}(W_Q, W_A, Z_Q, Y_A, Z_A, \theta_Q, \phi, \beta_Q, \beta_A | \alpha_\theta, \alpha_{\beta_Q}, \alpha_{\beta_A}, \alpha_\phi) = \\
&\left(\prod_{i=1}^{N_Q} \mathrm{P}(\theta_{Q_i}|\alpha_\theta) \left(\prod_{n=1}^{N_{Q_i}} \mathrm{P}(Z_{Q_i}^n|\theta_{Q_i}) \mathrm{P}(W_{Q_i}^n|Z_{Q_i}^n, \beta_Q) \right) \right) \\
&\prod_{i=1}^{N_Q} \prod_{j=1}^{N_{A_{Q_i}}} \prod_{n=1}^{N_{A_{i,j}}} \mathrm{P}(Y_{A_{i,j}}^n|\theta_{Q_i}) \mathrm{P}(Z_A^n|Y_{A_{i,j}}^n, \phi) \mathrm{P}(W_{A_{i,j}}^n|Z_{A_{i,j}}^n, \beta_A) \\
&\prod_{k=1}^{K} \mathrm{P}(\beta_Q^k|\alpha_{\beta_Q}) \prod_{l=1}^{L} \mathrm{P}(\beta_A^l|\alpha_{\beta_A}) \prod_{k=1}^{K} \mathrm{P}(\phi_k|\alpha_\phi)
\end{aligned} \tag{3.2}$$

where Z_Q, Y_A, Z_A, θ_Q, ϕ, β_Q, and β_A are the hidden variables and the observations, W_Q and W_A, are given over the entire corpus. The next step is to do inference and estimate the parameters of the model. This procedure estimates the posterior probability of the latent variables given the input parameters and observations. It is explained in detail in the next section. Inference can also be considered as an optimization process where the likelihood of the entire corpus of questions and answers

is optimized with respect to the hyperparameters $\theta_Q, \phi, \beta_Q, \beta_A$ given the data W_Q, W_A:

$$\mathrm{L}(\theta_Q, \phi, \beta_Q, \beta_A | W_Q, W_A) \triangleq \mathrm{P}(W_Q, W_A | \theta_Q, \phi, \beta_Q, \beta_A) = \prod_{i=1}^{N_Q} \mathrm{P}(W_{Q_i} | \theta_{Q_i}, \phi, \beta_Q, \beta_A) \prod_{i=1}^{N_Q} \prod_{j=1}^{N_{A_{Q_i}}} \mathrm{P}(W_{A_{i,j}} | \theta_{Q_i}, \phi, \beta_Q, \beta_A) \tag{3.3}$$

Algorithm 1: Generative process for Question Answering Topic Model

for *all question topics* k *in* $[1, K]$ **do**
  sample question mixture components $\vec{\beta_Q^k} \sim \mathrm{Dir}(\vec{\alpha_{\beta_Q}}, V)$;

for *all answer topics* l *in* $[1, L]$ **do**
  sample answer mixture components $\vec{\beta_A^l} \sim \mathrm{Dir}(\vec{\alpha_{\beta_A}}, V)$;

for *all question topics* k *in* $[1, K]$ **do**
  sample $\vec{\phi_k} \sim \mathrm{Dir}(\vec{\alpha_\phi}, L)$;

for *all questions* $Q_i \in [1, N_Q]$ **do**
  sample mixture proportion $\vec{\theta_{Q_i}} \sim \mathrm{Dir}(\vec{\alpha_\theta}, K)$;
  sample question length $N_{Q_i} \sim \mathrm{Poiss}(\xi)$;
  for *all words* $n \in [1, N_{Q_i}]$ *in question* Q_i **do**
    sample question topic index $Z_{Q_i}^n \sim \mathrm{Mult}(\vec{\theta_{Q_i}})$;
    sample term for word $W_{Q_i}^n \sim \mathrm{Mult}(\vec{\beta_Q^{Z_{Q_i}^n}})$;
  for *all answers* $A_{i,j} \in [1, N_{A_{Q_i}}]$ **do**
    sample answer length $N_{A_{i,j}} \sim \mathrm{Poiss}(\xi)$;
    for *all words* n *in* $[1, N_{A_{i,j}}]$ *in answer* $A_{i,j}$ **do**
      sample question topic index $Y_{A_{i,j}^n} \sim \mathrm{Mult}(\vec{\theta_{Q_i}})$;
      sample answer topic index $Z_{A_{i,j}}^n \sim \mathrm{Mult}(\vec{\phi_{Y_{A_{i,j}}^n}})$;
      sample term for word $W_{A_{i,j}}^n \sim \mathrm{Mult}(\vec{B_A^{Z_{A_{i,j}}^n}})$;

N_Q	number of Questions to generate.
$N_{A_{Q_i}}$	number of Answers to generate (question specific)
K	number of Q-topics.
L	number of A-topics.
V	number of terms t in vocabulary.
α_θ	hyperparameter on the mixing proportions (K-vector or scalar if symmetric).
α_ϕ	hyperparamter.
α_{β_A}	hyperparameter on the mixing components.
α_{β_Q}	hyperparameter on the mixing components.
θ	($N_Q \times K$ matrix)
ϕ	($K \times L$ matrix)
β_Q	($K \times V$ matrix)
β_A	($L \times V$ matrix)
N_{Q_i}	question length (question specific), here modeled with a Poisson distribution, with constant parameter ξ.
$N_{A_{i,j}}$	answer length (answer specific), here modeled with a Poisson distribution, with constant parameter ξ.
$Z^n_{Q_i}$	mixture indicator that chooses the Q-topic for the nth word in question Q_i.
$Y^n_{A_{i,j}}$	mixture indicator that chooses the Q-topic for the nth word in answer $A_{i,j}$.
$Z^n_{A_{i,j}}$	mixture indicator that chooses the A-topic for the nth word in answer $A_{i,j}$.
$W^n_{Q_i}$	term indicator for the nth word in question Q_i.
$W^n_{A_{i,j}}$	term indicator for the nth word in answer $A_{i,j}$.

Figure 3.3: Quantities in Question Answering Topic Model.

3.2 Inference and Parameter Estimation

In this section, we present a Gibbs sampling procedure for doing inference in the proposed model. The procedure estimates the parameters of our model, and identifies the structure posited in our data collection. For doing inference, we need to compute the posterior probability of the latent variables Z_Q, Y_A, Z_A, θ_Q, ϕ, β_Q, and β_A given the input parameters $\alpha_\theta, \alpha_{\beta_Q}, \alpha_{\beta_A}, \alpha_\phi$ and observations W_Q and W_A:

$$\begin{gathered}\mathrm{P}(Z_Q, Y_A, Z_A, \theta_Q, \phi, \beta_Q, \beta_A | W_Q, W_A, \alpha_\theta, \alpha_{\beta_Q}, \alpha_{\beta_A}, \alpha_\phi) = \\ \frac{\mathrm{P}(Z_Q, Y_A, Z_A, \theta_Q, \phi, \beta_Q, \beta_A, W_Q, W_A | \alpha_\theta, \alpha_{\beta_Q}, \alpha_{\beta_A}, \alpha_\phi)}{\mathrm{P}(W_Q, W_A | \alpha_\theta, \alpha_{\beta_Q}, \alpha_{\beta_A}, \alpha_\phi)}\end{gathered} \tag{3.4}$$

However, exact inference for the posterior is intractable. Gibbs sampling is an approximate inference algorithm. To emulate high dimensional probability distributions, Gibbs sampling constructs a Markov chain. Each dimension of the probability distribution is then sampled one at a time, conditioned on the values of all other dimensions. The Markov chain converges to the posterior distribution of the latent variables and the results can then be used to approximate the parameters of the model. Collapsed Gibbs sampling [13] is a variation of Gibbs sampling where one or more parameters are integrated out when sampling other variables. We use collapsed Gibbs sampling to sample the variables Z_Q, Y_A, and Z_A, integrating out θ_Q, ϕ, β_Q, and β_A.

For QATM, we sample Y_A, and Z_A jointly and Z_Q separately. We need to compute two conditional distributions $\mathrm{P}(Z^n_{Q_i} | Z^{-n}_Q, Y_A, Z_A, W_Q, W_A)$ and $\mathrm{P}(Y^n_{A_{i,j}}, Z^n_{A_{i,j}} | Y^{-n}_{A_{i,j}}, Z^{-n}_{A_{i,j}}, Z_Q, W_Q, W_A)$ where $Z^n_{Q_i}$ represents Q-topic assignment for word n in question Q_i and Z^{-n}_Q denotes Q-topic assignments for all other words except the current word $W^n_{Q_i}$. Moreover, $Y^n_{A_{i,j}}$ denotes the Q-topic assignment for word n in answer j of question i and $Z^n_{A_{i,j}}$ represents the A-topic assignment for the same word conditioned on $Y^n_{A_{i,j}}$. We have:

$$\begin{gathered}\mathrm{P}(Z^n_{Q_i} = k | Z^{-n}_Q, Y_A, Z_A, W_Q, W_A) \propto \\ \frac{\alpha_{\beta_Q} + C_Q^{kW^n_{Q_i}}}{\sum_{v=1}^{V}(\alpha_{\beta_Q} + C_Q^{kv})} \frac{\alpha_\theta + C^k_{Q_i} + C^k_{A_{Q_i}}}{\sum_{i'=1}^{k}(\alpha_\theta + C^{i'}_{Q_i} + C^{i'}_{A_{Q_i}})}\end{gathered} \tag{3.5}$$

where C^{kv}_Q is the number of times word v is assigned to Q-topic k. Moreover, $C^k_{Q_i}$ is

the number of times Q-topic k is assigned to words in question Q_i and $C^k_{A_{Q_i}}$ denotes the number of times A-topics for words in the set of answers for question Q_i are drawn conditioned on Q-topic k.

$$\begin{aligned}&\mathrm{P}(Y^n_{A_{i,j}} = k, Z^n_{A_{i,j}} = l | Y^{-n}_{A_{i,j}}, Z^{-n}_{A_{i,j}}, Z_Q, W_Q, W_A) \propto \\ &\frac{\alpha_{\beta_A} + C_A^{lW^n_{A_{i,j}}}}{\sum_{v=1}^{V}(\alpha_{\beta_A} + C_A^{lv})} \frac{\alpha_\theta + C^k_{Q_i} + C^k_{A_{Q_i}}}{\sum_{i'=1}^{k}(\alpha_\theta + C^{i'}_{Q_i} + C^{i'}_{A_{Q_i}})} \frac{C^l_k + \alpha_\phi}{\sum_{i=1}^{L}(\alpha_\phi + C^i_k)}\end{aligned} \tag{3.6}$$

where C^{lv}_A is the number of times word v is assigned to A-topic l. Moreover, C^l_k is the number of times an A-topic l is drawn conditioned on a Q-topic k in the entire corpus. More detailed derivations are given in Appendix A.

Algorithms 2, 3, and 4 show the inference procedure for our model. The Gibbs sampling algorithm is initialized by assigning each question word token to a random Q-topic in [1..K] and each answer word token to a random Q-topic [1..K] and random A-topic [1..L]. A number of initial samples (referred to as burn-in samples) are disregarded to eliminate the influence of initialization parameters. However, the subsequent samples start to approximate the target distribution and a subset of them are reserved as representatives from the distribution. To draw independent samples and prevent correlations between them, this subset is sampled at regularly spaced intervals.

Algorithm 2: QATM General

Data: word vectors or observations $\{\vec{W_Q}\}$ and $\{\vec{W_A}\}$, hyperparameters α_θ, α_{β_Q},α_{β_A},α_ϕ, Q-topic number K, A-topic number L

Result: Q-topic associations $\{\vec{Z_Q}\}$, $\{\vec{Y_A}\}$, A-topic associations $\{\vec{Z_A}\}$, multinomial parameters θ_Q, ϕ, β_Q, and β_A, hyperparameter estimates α_θ, α_{β_Q}, α_{β_A}, α_ϕ

```
// initialization
```

$QATMInitializeCountVariables()$;

```
// Gibbs sampling over burn-in period and sampling period
```

$QATMGibbsSampling()$;

if *converged and L sampling iterations since last read out* **then**

```
// different parameters read out are averaged
```

read out parameter sets;

Algorithm 3: QATM Initialize Count Variables

zero all count variables C_Q^{kv}, C_A^{lv}, $C_{Q_i}^k$, $C_{A_{Q_i}}^k$, C_k^l, C_k, C_l, $SUMC_k^l$;

for *all questions* Q_i *in* $[1, N_Q]$ **do**

  for *all words* n *in* $[1, N_{Q_i}]$ *in question* Q_i **do**

    sample Q-topic index $Z_{Q_i}^n = k \sim \text{Mult}(1/K)$;

    increment Q-topic-term count: $C_Q^{kW_{Q_i}^n} += 1$;

    increment Q-topic-term sum: $C_k += 1$;

    increment question-Q-topic count: $C_{Q_i}^k += 1$;

  for *all answers* $A_{i,j}$ *in* $[1, N_{A_{Q_i}}]$ **do**

    for *all words* n *in* $[1, N_{A_{i,j}}]$ *in answer* $A_{i,j}$ **do**

      sample Q-topic index $Y_{A_{i,j}^n} = k \sim \text{Mult}(1/K)$;

      sample A-topic index $Z_{A_{i,j}^n} = l \sim \text{Mult}(1/L)$;

      increment A-topic-term count: $C_A^{lW_{A_{i,j}}^n} += 1$;

      increment A-topic-term sum: $C_l += 1$;

      increment answer-Q-topic count: $C_{A_{Q_i}}^k += 1$;

      increment Q-topic-A-topic count $C_k^l += 1$;

      increment Q-topic-A-topic sum: $SUMC_k^l += 1$;

Algorithm 4: QATM Gibbs Sampling

while *not finished* **do**

  for *all questions* Q_i *in* $[1, N_Q]$ **do**

    for *all words* n *in* $[1, N_{Q_i}]$ *in question* Q_i **do**

      // for the current assignment of k to a term n for word $W^n_{Q_i}$:

      decrement counts and sums: $C_Q^{kW^n_{Q_i}} -= 1, C_k -= 1, C^k_{Q_i} -= 1$;

      // multinomial sampling acc. to Eq. 3.5 (decrements from the previous step)

      sample Q-topic index $\tilde{k} \sim \mathrm{P}(Z^n_{Q_i} = k | Z^{-n}_Q, Y_A, Z_A, W_Q, W_A)$;

      // for the new assignment of $\tilde{k}$ to the term n for word $W^n_{Q_i}$

      increment counts and sums: $C_Q^{\tilde{k}W^n_{Q_i}} += 1, C_{\tilde{k}} += 1, C^{\tilde{k}}_{Q_i} += 1$;

    for *all answers* $A_{i,j}$ *in* $[1, N_{A_{Q_i}}]$ **do**

      for *all words* n *in* $[1, N_{A_{i,j}}]$ *in answer* $A_{i,j}$ **do**

        // for the current assignment of k and l to a term n for word $W^n_{A_{i,j}}$:

        decrement counts and sums: $C_A^{lW^n_{A_{i,j}}} -= 1, C_l -= 1, C^k_{A_{Q_i}} -= 1, C^l_k -= 1, SUMC^l_k -= 1$;

        // multinomial sampling acc. to Eq. 3.6 (decrements from the previous step)

        sample Q-topic index and sample A-topic index $\tilde{k}, \tilde{l} \sim \mathrm{P}(Y^n_{A_{i,j}} = k, Z^n_{A_{i,j}} = l | Y^{-n}_{A_{i,j}}, Z^{-n}_{A_{i,j}}, Z_Q, W_Q, W_A)$;

        // for the new assignment of $\tilde{k}$ and $\tilde{l}$ to the term n for word $W^n_{A_{i,j}}$

        increment counts and sums: $C_A^{\tilde{l}W^n_{A_{i,j}}} += 1, C_{\tilde{l}} += 1, C^{\tilde{k}}_{A_{Q_i}} += 1, C^{\tilde{l}}_{\tilde{k}} += 1, SUMC^{\tilde{l}}_{\tilde{k}} += 1$;

3.3 Approximate Inference Algorithm Evaluation

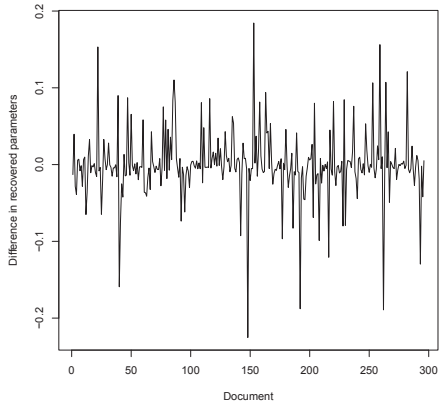
To examine the approximate inference algorithms ability in recovering true model parameters, we used the model to simulate a small dataset and applied the inference algorithm. Our simulated dataset had three Q-topics ($K = 3$) and four A-topics ($L = 4$). These statistics were chosen so that validation process was simplified and results were interpretable. For the simulated dataset, the true Q-topic distribution, represented by θ, has three components θ_1, θ_2, and θ_3. After applying the inference algorithm, this distribution is estimated for each document, and is represented by $\theta\prime$, with the components $\theta\prime_1, \theta\prime_2$, and $\theta\prime_3$. The difference between corresponding components of the true Q-topic distribution and the retrieved Q-topic distribution is presented in Figure 3.4. The small differences show that the approximate inference algorithm is capable of retrieving the actual parameters from a dataset that is generated by the model.

3.4 Model Analysis

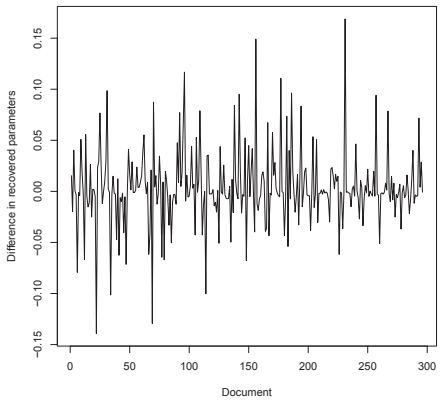
QATM models the content of CQA archives and extracts their topical structure. This structure is represented by soft associations between the hidden topics and the observed variables. To analyze the predicted structure, we used it for the tasks of Question Answering and Automatic Tagging.

3.4.1 Question Answering

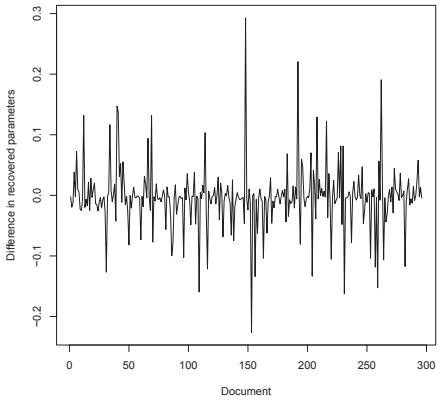
We use the model for the task of Question Answering, in which existing question-answer pairs in the archive are automatically retrieved and ranked given a newly submitted question. To retrieve documents similar to a query, the topics associated with the query can be compared against the topics associated with the documents in the collection [13]. In particular, the topic distribution of the query is estimated using the model and the trained topic distributions. A similarity measure is then utilized to compare the query topic distribution with the topic distributions of individual documents in the collection. Kullback-Leibler (KL) divergence [19] is a similarity



(a) $\theta_1 - \theta\prime_1$



(b) $\theta_2 - \theta\prime_2$



(c) $\theta_3 - \theta\prime_3$

Figure 3.4: Vector difference of the true Q-topic distribution(θ) and the retrieved Q-topic distribution($\theta\prime$).

measure that is defined between two discrete random variables X and Y, as:

$$\mathrm{D}_{KL}(X||Y) = \sum_{n=1}^{N} p(X=n)[\log_2 p(X=n) - \log_2 p(Y=n)] \tag{3.7}$$

However, KL divergence is not a symmetric measure. We have used the Jensen-Shannon (JS) distance, a symmetrised extension of KL divergence as our similarity measure:

$$\mathrm{D}_{JS}(X||Y) = \frac{1}{2}[\mathrm{D}_{KL}(X||M) + D_{KL}(Y||M)] \tag{3.8}$$

where $M = \frac{1}{2}(X+Y)$. This similarity measure produces a ranking over the documents in the collection [13].

3.4.2 Automatic Tagging

Tagging is the process of annotating a resource with metadata or keywords that describe it. Tags help categorize content and can therefore be used to browse and organize electronic archives. The QATM model provides a soft clustering of the documents and of the terms of a collection by associating them to topics. In particular, the mixed membership vector estimated for each document, assigns topics or tags to documents. To evaluate the predicted clustering we use the clustering performance measures introduced in [27] and compare the clustering performance of the model against a gold standard extracted from Stack Overflow. We describe the details of this procedure and present experimental results in the following chapter.

Chapter 4

Experimental Study

In the previous chapter, we presented a new topic model for the content of CQA archives. The model is designed to explicitly capture topic dependency and correlation between questions and answers, and model differences in their vocabulary. To analyze this model, we utilize it for the tasks of Question Answering and Automatic Tagging.

In this chapter we present experimental results that evaluate the model's performance against a dataset extracted from the programming website, Stack Overflow. We compare QATM's performance with a keyword based model, TFIDF and a topic model, LDA.

4.1 Dataset

4.1.1 Stack Overflow

We evaluate our model using a real world dataset extracted from Stack Overflow [1]. This is a programming Question and Answering (Q & A) website, where developers can share technical information amongst themselves. To maintain an archive of high quality questions and answers, Stack Overflow employs popularity voting and allows users to vote upon and edit questions and answers. The users' contribution to the website is represented by reputation points and badges, based upon which they are granted more moderation capabilities and permissions. An archive of the content of this website is released every two months. For our experiments, we used the January 2011 data dump. Some statistics of this data are given in Table 4.1. Appendix B depicts a summary of the main entities on Stack Overflow.

In the next section, we describe the dataset creation procedure. This data is used to evaluate QATM model for both the Question Answering task and the Automatic Tagging task.

[1] `http://stackoverflow.com/`

#Questions	1,188,585
#Answers	2,939,767
#Tags	27,659
#Users	440,672
Avg #answer per question	2.4818
Avg #tags per question	2.9254
Avg score of questions	1.4967
Avg score of answers	1.8478

Table 4.1: Stack Overflow statistics, January 2011

4.1.2 Train Data

When a question is posted on Stack Overflow, it is tagged with labels or tags. To extract a representative subset of questions and answers from the large archive available on this website, we initially extracted a smaller subset of tags. We examined tag frequency and tag co-occurrence statistics, as shown in Figure 4.1, and identified three criteria for tag selection:

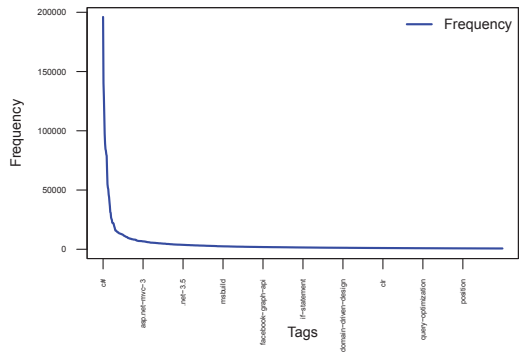
1. To ensure that there would be sufficient data available for training the model, popular tags were required.

2. To maintain a similar tag distribution as the original data collection, and construct a realistic dataset, tag overlap (or tag co-occurrence) had to be preserved.

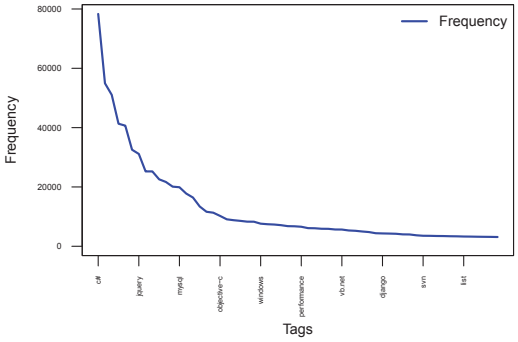
3. However, to ensure the model could uncover challenging patterns in the data, we required a set of conceptually overlapping tags and a set of distinctly different tags with distinguishable keywords.

With regard to the three criteria, we manually selected a total of 21 tags. We chose 7 tags that were popular and that co-occurred frequently with other tags, 7 tags that were popular but co-occurred less frequently, and 7 tags that were popular and rarely co-occurred with other tags. The selected tags are shown in Table 4.2. Subsequently, for each tag we randomly collected 200 questions (4200 questions in total).

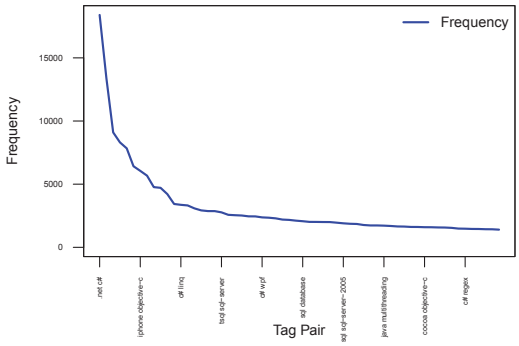
The content on Q&A websites is user-generated and therefore, numerous questions and answers with varying quality levels have been created. However, on Stack Overflow, users score posts based on their relevance and correctness. As a result,



(a) Tag frequency of top 1000 tags



(b) Frequency of top 60 tags, from posts with more than two answers



(c) Pairwise frequency of top 60 tags, from posts with more than two answers

Figure 4.1: Tag frequencies.

correctly posed questions or more relevant answers usually have higher scores. To allow the model to correctly learn the topic dependencies of a question and its answers, we extracted the 4 most relevant answers for each question using the scores given to answers by users based on their perceived relevance and correctness. At the end of this step we had extracted 15822 question-answer pairs for the train dataset[2]. Appendix C.1 describes a summary of the train data creation steps.

Our model requires information about the answers of a question. Hence, to create the training set for our model, we merged individual questions with their relevant answers into one document. Therefore each document consists of a question and

[2]Some of the questions had less than 4 answers. Furthermore, the 15822 question-answer pairs correspond to 4184 questions. There are two reasons for the decrease in the number of QAs from 4200 to 4184: 1) Around 12 of the Questions were invalid, meaning that they were identified as duplicates of other questions that existed in our train data, so these duplicates were removed (4200 -12 = 4188) 2) 4 of the questions selected in the second step did not have answers and so were dropped, that left us with 4184 valid QAs.

C#	.net	Sql
Sql-server	Java-script	css
Java	Ruby	Ruby-on-rails
Wpf	iphone	Web-development
Android	Windows	Delphi
Django	Python	C
Bash	Linux	homework

Table 4.2: Subset of 21 selected tags

at most four of its most relevant answers. However, because the model distinguishes between questions and answers, it is capable of retrieving and ranking question answer pairs.

To retrieve and rank question-answer pairs, LDA and TFIDF require that documents in their training set consist of question answer pairs. Hence, for these models, we have used the same information to create a training set in which a document consist of a single question answer pair.

4.1.3 Test Set

Question Answering

To evaluate the answer retrieval performance of our model, we extracted a set of questions from Stack Overflow referred to as *duplicates*. These are questions that are similar to one or more existing questions in the archive but use different words and structure. Because they increase the archive's redundancy, duplicates are considered as a negative feature of CQA websites. Therefore, Stack Overflow users identify, vote upon and close such questions.

However, there are also benefits to having alternates for the same question. Users often tend to search the archive using different terminology; duplicates increase visibility and make it more likely for users to find relevant information. From the viewpoint of our research however, duplicates are extremely important because they provide multiple variants for a question. We create a gold standard set from these duplicates and incorporate them into our evaluation paradigm. Appendix C.2 gives a detailed description of the duplicate extraction process.

Without duplicates, a possible methodology for evaluating the models performance, would have been to conduct a user study. However, there are some disadvantages with user studies. In particular, they are expensive to conduct, both in terms of user's time and compensations. A significant amount of time and effort is required to carefully consider various components of the study and conduct successful user studies. Experiments have to be designed so that they provide robust answers to the questions of interest and lead to statistically significant results, the participants have to be chosen so that they represent the true population of users, future information needs have to be predicted, and important decisions must be made based on vague expectations. In addition analyzing and interpreting experimental results can be hard. Typically, the outcome of studies can only answer small questions, and larger conclusions cannot be drawn.

Duplicates enabled us to incorporate perceptions of the end-users of the technology into the evaluation without the need to conduct a user study. They also represent natural information seeking behavior which is important because the results are real world and can lead to valid generalizations.

On Stack Overflow three classes of duplicate questions have been identified and different handling mechanisms have been defined [3]:

1. Cut-and-paste duplicates: These questions are the exact copy of a previous question. They are the very definition of exact duplicates. This group of duplicate questions are voted down by users and flagged for moderator attention. They are then deleted from the system by the moderators.

2. Accidental duplicates: These questions are semantically similar but lexically different from a previously asked question in the archive. They have the same answer. Stack Overflow users identify, vote upon and close them as "exact duplicates" of another question. They link these duplicates to the original question by posting the URL of the original question as a comment or edit in

[3]Refer to:

1. `http://blog.stackoverflow.com/2009/04/handling-duplicate-questions/`
2. `http://meta.osqa.net/questions/1086/`
3. `http://blog.stackoverflow.com/2009/05/linking-duplicate-questions/`

the duplicate.

3. Borderline duplicates: The questions in this set cover the same grounds as a previous question in the archive, however their overlap with those questions is ambiguous. This means that their uniqueness and commonality is subject to interpretation. These questions are tagged by users of Stack Overflow so that they naturally group with other relevant questions.

We extracted a set of accidental duplicates for questions in our train data and created a gold standard set. Examples of accidental and borderline duplicates are given in Appendix C.3.

Automatic Tagging

We evaluate the clustering quality of our model against a *gold standard* available in the data on the website. This gold standard consists of the tags assigned to questions in our training dataset by users of Stack Overflow.

4.1.4 Dataset Preprocessing

Both the training and test datasets were preprocessed to remove HTML markup, numbers, punctuation, signs such as '@, %, !,'. In addition, the documents were stemmed (Porter stemmer [4]) and stop-words were removed (using Mallet [21])[5].

4.1.5 Dataset Tag Statistics

Figure 4.2 compares the distribution of tags in the training and the test set. The figure shows that the tag selection procedure has resulted in balanced classes of tags in the training data. In addition, even though there are some discrepancies between the test and train set tags, they generally follow the same probability distribution (the data samples come from the same statistical populations).

[4] `http://tartarus.org/martin/PorterStemmer/`

[5] Two options were used for mallet, –keep-sequence and remove-stopwords

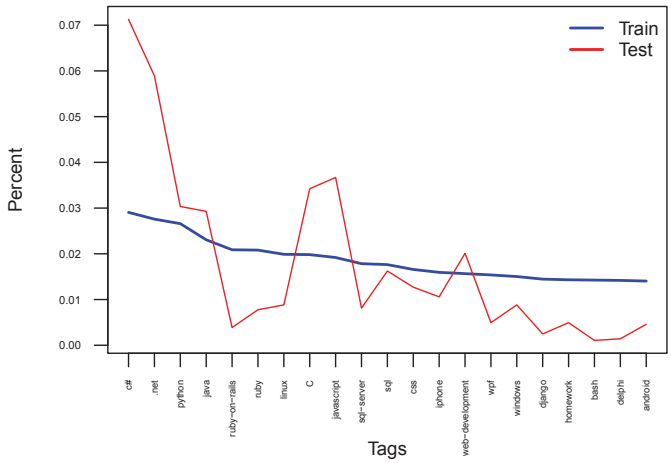


Figure 4.2: Tag distribution in train set and test set.

4.2 Baselines

4.2.1 TFIDF

TFIDF is a common term weighting scheme. It can be used to assign weights to terms in the vector space model. The TFIDF weight for a term has two components, the term frequency and the inverse document frequency. Term frequency of term t in document d is denoted by $TF(t, d)$. It is the number of times term t appears in document d. This component is normalized to prevent bias towards longer documents. The inverse document frequency of term t in corpus C is denoted by $IDF(t, C)$. It describes the general importance of a term in a document collection. Altogether the TFIDF weight for a term is:

$$\begin{aligned} \text{TFIDF}(t, d, C) &= \text{TF}(t, d) \times \text{IDF}(t, C) \\ &= \text{TF}(t, d) \times \log \frac{|C|}{1 + |\{d \in C : t \in d\}|} \end{aligned} \tag{4.1}$$

where $|C|$ is the size of the train corpus. $|\{d \in C : t \in d\}|$ is the number of documents in the train corpus where the term t appears. The TFIDF value increases proportionally to the frequency of a term in a document, but is counterbalanced by the frequency of the term in the whole corpus.

Given two objects a and b, their similarity can be defined as the cosine of the angle between their vector representations (i.e. v_a and v_b):

$$\text{Cosine Similarity}(v_a, v_b) = \frac{v_a \cdot v_b}{||v_a||||v_b||} \tag{4.2}$$

In this work, TFIDF is used as a representative of word-based approaches to the question answering task. TFIDF vectors are obtained for the train data. Using IDF values obtained for terms in the train data, TFIDF vectors are computed for the queries in the ground truth. Subsequently, for each of these queries, documents in the train data are ranked using cosine similarity between their vector representations.

4.2.2 LDA

LDA is a widely used topic model. It treats words in a document as multinomial observations. For each document, a distribution over topics is estimated where each topic is a distribution over the words in the corpus. The derived topics allow the model to map the high dimensional literal representations into a lower dimensional topic space. The similarity between documents and queries can then be computed at a semantic level using the distance between documents and queries in this topical space.

The graphical model for LDA is given in Figure 4.3. Essentially, to generate words in a given document m, the generative process of LDA assumes to first draw a topic Z_m^n from the document-specific topic distribution θ_m. Then, the corresponding topic-specific term distribution $\phi_{Z_m^n}$ is used to draw a word. This procedure is repeated for each word in the document.

In this work, LDA is applied to the data to obtain semantic-level representations of the corpus. We used the LDA implementation in Mallet [21]. For each query in the ground truth, documents in the train data are ranked according to the procedure described in 3.4.1.

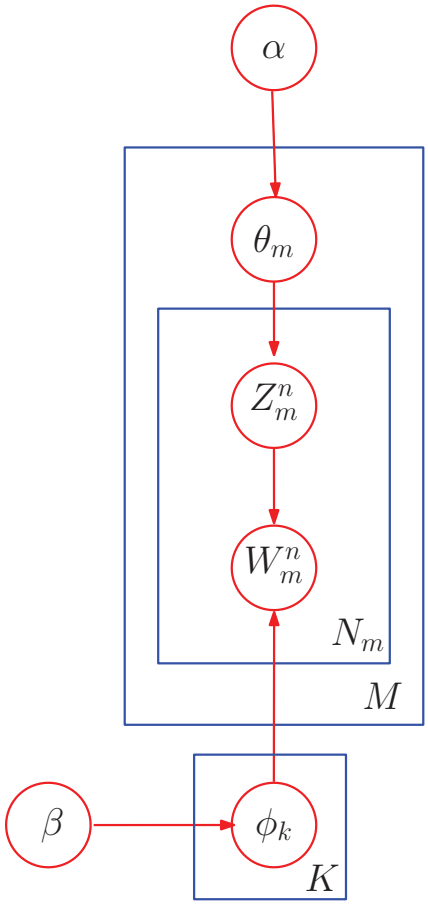


Figure 4.3: Latent Dirichlet Allocation (LDA) Topic Model. The simple intuition behind this model is that documents exhibit multiple topics.

4.3 Evaluation

4.3.1 Question Answering

We compare our model with LDA and TFIDF in terms of retrieving the right answer for a given query and report TopN and Mean Average Precision (MAP) performance measures:

- **Mean Reciprocal Rank (MRR):** Reciprocal Rank for a query is the reciprocal rank of the first relevant answer. The mean reciprocal rank for a set of queries [28] is the average of their reciprocal ranks. TopN for a set of queries, considers a cut-off N and measures the reciprocal rank for each query and the mean reciprocal rank for the set of queries:

$$MRR(Q) = \frac{1}{|Q|} * \sum_{q=1}^{Q} \frac{1}{rank_q} \tag{4.3}$$

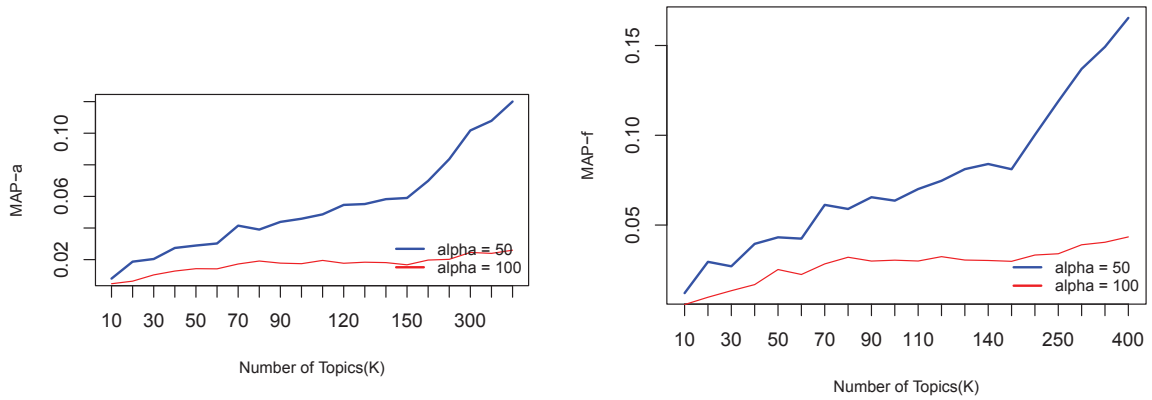
  where Q is the set of queries and $rank_q$ is the rank of the first relevant document for query q.

- **Mean Average of Precision (MAP):** Since each duplicate can have multiple correct answers (each query question can have several relevant answers), a measure that considers the order of the correct answers and emphasizes ranking them higher, is needed. Average precision for a query, is the average of the precisions computed at the point of each of the relevant answers in the ranked list:

$$AveP(Q) = \frac{\sum_{rank=1}^{N}(P(rank) * rel(rank))}{\text{Number of relevant documents}} \tag{4.4}$$

  where $rank$ is a given rank in the sequence, N the number retrieved, $rel(rank)$ is a binary function on the relevance of a given rank, and $P(rank)$ is precision at a given cut-off rank. Mean average precision for a set of queries is the mean of the average precision scores for each query:

$$MAP(Q) = \frac{1}{|Q|} * \sum_{q=1}^{Q} AveP(q) \tag{4.5}$$



(a) LDA MAPa (b) LDA MAPf

Figure 4.4: LDA configuration.

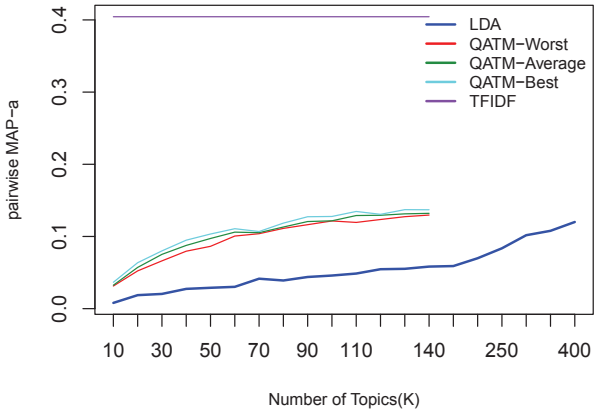
In this work we report two types of MAP, MAPf and MAPa. This is because each duplicate can have several correct questions in the train data. Each question has at most four answers, hence each duplicate has several relevant question answer pairs. To calculate MAPf (MAPfirst) we group the question-answer pairs by distinct questions and return the first ranked question-answer pair in each group. To calculate MAPa (MAPall) for each duplicate we simply take the average of all the question-answer pairs without grouping them.

LDA Configuration

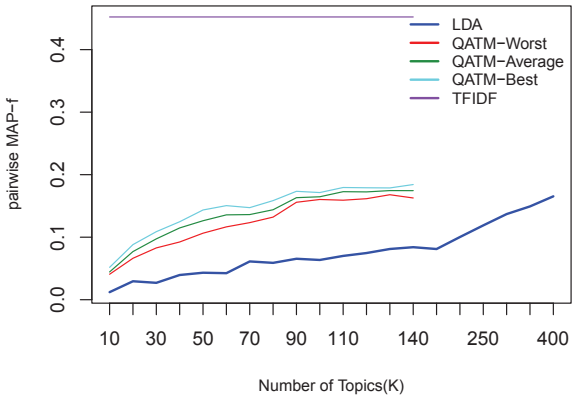
The Dirichlet hyperparameters in LDA generally denote the belief about observations. They control the sparsity of the mulitominal distributions. Lower values of these parameters results in sparse distributions and higher values result in dense distributions. Good model quality has been reported for $\alpha = 50/k$ and $\beta = 0.01$. To find the optimal value of these parameters for our dataset, we conducted two sets of experiments with $\alpha = 50/k$ and $\alpha = 100/k$ and found the former results in the best performance. Figure 4.4.a shows MAPa results and Figure 4.4.b shows MAPf results.

Results

We compare the retrieval performance of our model with LDA and TFIDF. Results for MAP at various numbers of topics are plotted in Figures 4.5.a and 4.5.b. Our model has two sets of topics, Q-topics and A-topics. Since the Q-topics in our model



(a) MAPa

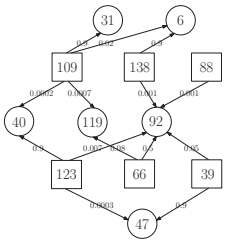


(b) MAPf

Figure 4.5: MAP results for question-answer pair retrieval.

	Top1	Top2	Top3	Top4	Top5
LDA	0.038	0.054	0.059	0.063	0.066
QATM-Worst	0.108	0.127	0.138	0.144	0.148
QATM-Average	0.122	0.142	0.151	0.156	0.16
QATM-Best	0.131	0.152	0.161	0.164	0.168
TFIDF	0.363	0.395	0.417	0.426	0.433

Table 4.3: TopN results for question-answer pair retrieval



Topic Dependencies	Q-topic 39	Q-topic 66	Q-topic 88	Q-topic 109	Q-topic 123	Q-topic 138	A-topic 6	A-topic 31	A-topic 40	A-topic 47	A-topic 92	A-topic 119
	connect	end	java	email	stream	http	http	part	stream	connect	thi	answer
	server	begin	scala	piec	buffer	googl	www	email	buffer	socket	wai	question
	close	procedur	lang	send	mp	www	googl	send	data	port	ani	point
	port	assign	jvm	contact	audio	blog	org	home	length	network	onli	case
	open	declar	run	split	call	site	html	target	byte	send	doe	order
	client	val	util	mail	plai	org	blog	work	close	server	make	actual
	servic	direct	org	input	data	stackoverflow	articl	contact	mp	data	someth	follow
	remot	dir	class	final	download	en	en	messag	flush	client	check	give
	network	var	jdk	subject	decod	html	link	mail	reader	receiv	exampl	ye
	socket	proc	swing	sm	sound	sampl	aspx	sourc	upload	remot	doesn	expect
	resourc	const	jrubi	separ	util	wiki	wiki	sm	audio	summari	code	sinc
	notif	record	vm	pop	flush	place	post	relpath	output	proxi	befor	assum
	possibl	integ	librari	millisecond	upload	link	stackoverflow	split	memorystream	close	wa	consid
	transact	tmat	api	msxml	close	twitter	doc	address	payment	conn	note	happen
	local	vehicleid	apach	receiv	clap	msdn	archiv	smtp	dest	timeout	find	otherwis
	support	tform	bytecod	attach	audiomanag	home	msdn	gmail	bankhead	param	possibl	howev
	sqlconnect	fd	jython	regist	programmat	forum	librari	receiv	input	packet	abov	side
	asynchron	dynam	ee	programmat	listen	articl	github	subject	result	open	don	correct
	transactionscop	xmpobject	awt	current	video	aspx	found	spam	assert	implement	problem	inform
	applic	dcanva	sun	previousposit	inputstream	archiv	amp	bodi	filepath	host	put	sai

Figure 4.6: Topical dependencies captured by QATM with examples of Q-topics and A-topics represented by their first 20 most probable words.

are similar to topics in the LDA model, when comparing to the LDA model, we report three MAP performance values for our model. Given the same number of topics for both models, we report an average, worst and best performance over a range of A-topic numbers for our model. These are denoted by QATM-Average, QATM-Worst and QATM-Best respectively. In addition, because of the computational costs of running our model (Appendix E), we only experimented with a maximum of 140 Q-topics and A-topics. But the lower computational costs for LDA allowed us to run experiments with a maximum of 400 topics. TFIDF is not a topic-based approach, and its performance is independent of the number of topics. The results show that our model performs significantly better than LDA. However, TFIDF outperforms both methods. Table 4.3 shows the TopN retrieval performance of the three models. Again, TFIDF performs better than the topic-based approaches, but our model outperforms the LDA model. This indicates that our model can be used in combination with other information retrieval methods for improving results.

Our model is capable of capturing topical dependencies between questions and answers. Examples of topics from a model trained with 140 Q-topics and 120 A-topics are shown in Figure 4.6. Each topic is represented by its first 20 most probable words. In addition, the graph in Figure 4.6 shows the dependencies discovered between topics in questions and answers.

Analysis of Results

Figure 4.7 shows the rank distribution for QATM and TFIDF, where the rank domain has been broken into intervals of 100. From this figure it can be seen that TFIDF retrieves a greater number of question-answer pairs in the top 100 results of each query. To understand TFIDF's superior performance, we assess the data. We define two metrics for measuring the lexical overlap between a query and a document (i.e in this case, a document consists of a question-answer pair). The first measure, *Proportion of Distinct Common Words over Distinct Query Words* (DCoDQ) is defined as:

$$\text{DCoDQ}(q, d) = \frac{|\text{Distinct Common Words}(q, d)|}{|\text{Distinct Words}(q)|} \tag{4.6}$$

where q is the query, and d is the document. The numerator is the the number of distinct common words between a query and a document, and the denominator is the number of distinct words in the query. The second measure, *Proportion of Length of Common Words over Length of Query Words* (LCoLQ) is defined as:

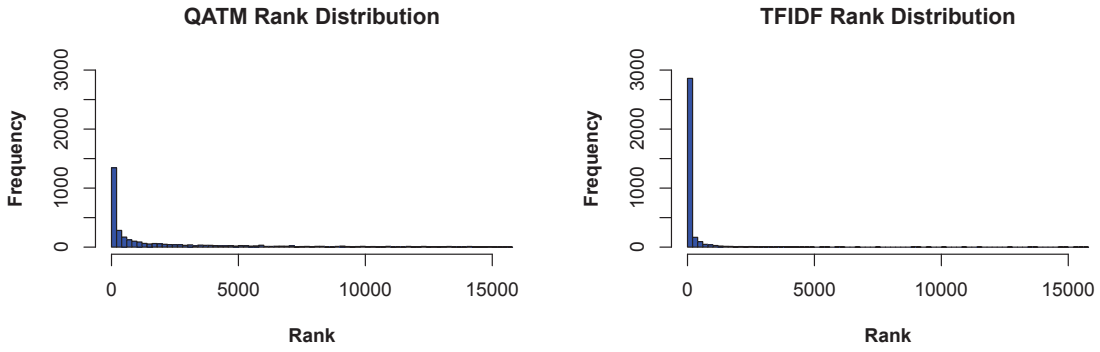
$$\text{LCoLQ}(q, d) = \frac{|\text{Common Words(q,d)}|}{|\text{Words}(q)|} \tag{4.7}$$

where q is the query, and d is the document. The numerator is the length (or number) of common words between a query and a document, and the denominator is the length of words in the query.

The 822 duplicates (queries) in our ground truth correspond to 3381 relevant question-answer pairs in the train data. Hence, for each data point consisting of a duplicate and a question-answer pair (3381 data points), we calculated DCoDQ and LCoLQ measures. Figure 4.8 plots the difference in ranks produced by QATM and TFIDF against the DCoDQ of each data point. Figure 4.9 plots the difference in ranks against the LCoLQ.

We also calculated the percentage of question-answer pairs that are retrieved in the top 100 results for their corresponding query. This percentage is 33.5% for QATM, 76% for TFIDF, and an overlap of 29.5%. Figures 4.8b and 4.9b denote the ranking difference against lexical similarity of these data items.

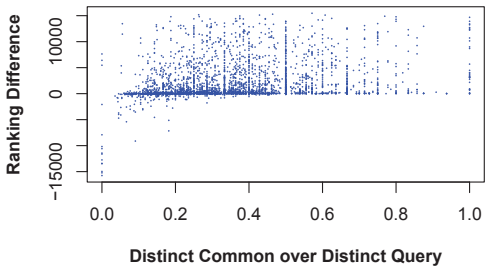
To conclude, the figures presented in this section indicate that our ground truth and train data are more lexially similar than semantically similar; this means that the nature of the dataset is such that it favors keyword-based methods such as TFIDF.



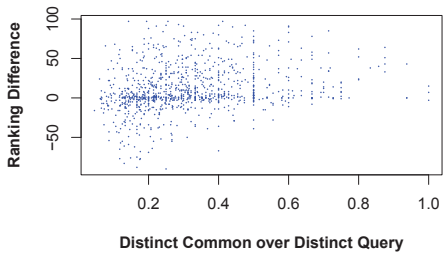
(a) QATM rank distribution

(b) TFIDF rank distribution

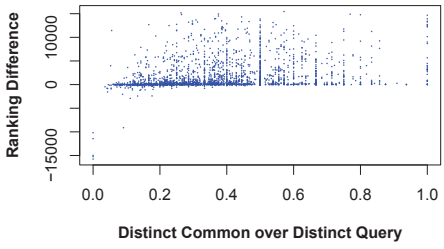
Figure 4.7: QATM and TFIDF rank distribution.



(a) All datapoints

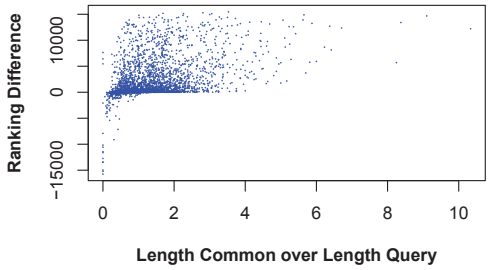


(b) Datapoints for which both methods produce rank less than 100

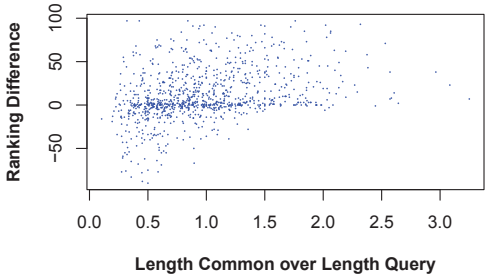


(c) Datapoints with at least one method producing rank less than 100

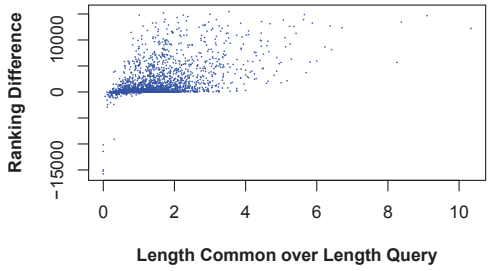
Figure 4.8: Ranking difference of datapoints against their lexical overlap (DCoDQ). Data points in the left of the diagrams correspond to query questions that do not have much word overlap with the retrieved question-answer pair.



(a) All datapoints



(b) Datapoints for which both methods produce rank less than 100



(c) Datapoints with at least one method producing rank less than 100

Figure 4.9: Ranking difference of datapoints against their lexical overlap (LCoLQ). Data points in the left of the diagrams correspond to query questions that do not have much word overlap with the retrieved question-answer pair.

4.3.2 Automatic Tagging

The QATM model provides a soft clustering of the documents and of the terms of a collection by associating them to topics. In particular, the mixed membership vector estimated for each document, assigns topics or tags to documents. We evaluate the clustering quality of our model based on a *gold standard* available in the data on the website. This gold standard consists of the tags assigned to questions in our dataset by users of Stack Overflow. To compare the tag distribution of individual documents with the mixed membership vector the model has estimated for the document, we use the performance metrics introduced in [27]. The proposed metrics are based on the concept of BCubed metrics [3]. BCubed metrics measure precision and recall for each data point. The precision of one data item is the proportion of items in the same cluster that belong to its category. Recall of a data item is the proportion of items from its category that appear in its cluster. The category of a data point is assigned by the gold standard. The cluster for a data point is assigned by the clustering algorithm. Aggregate precision and recall measures are obtained by averaging over all pairs of items. The harmonic mean of precision and recall is defined as the F-measure.

In the case of overlapping clusters [2], where there is a 0/1 assignment in which items can be assigned to multiple clusters, precision and recall are defined as:

$$Precision(e, e') = \frac{Min(|C(e), C(e')|, |L(e), L(e')|)}{|C(e), C(e')|} \tag{4.8}$$

$$Recall(e, e') = \frac{Min(|C(e), C(e')|, |L(e), L(e')|)}{|L(e), L(e')|} \tag{4.9}$$

where e and e' are two data items, $L(e)$ is the set of categories and $C(e)$ is the set of clusters assigned to e. $|C(e), C(e')|$ is the number of categories common to e and e'.

In the case of overlapping clusters, in which each item has a mixed membership vector assigned to it by the model and a true mixed membership vector, the metrics are extended as:

$$Precision(e, e') = \frac{Min(|\pi(e), \pi(e')|.|\gamma(e).\gamma(e')|)}{|\pi(e).\pi(e')|} \tag{4.10}$$

$$Recall(e, e') = \frac{Min(|\pi(e).\pi(e')|, |\gamma(e).\gamma(e')|)}{|\gamma(e).\gamma(e')|} \tag{4.11}$$

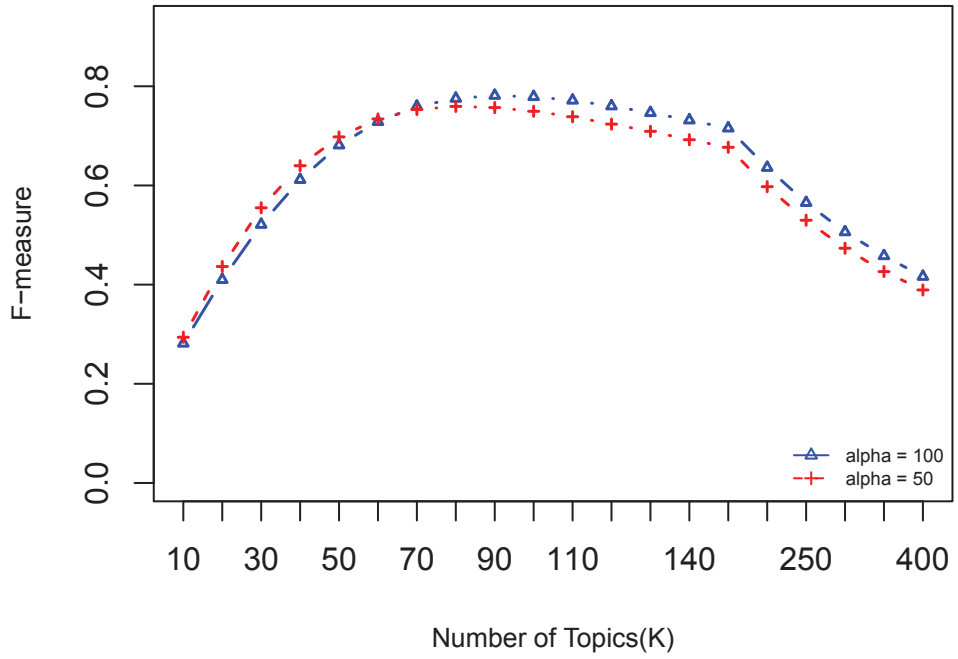


Figure 4.10: LDA configuration.

where $\pi(e)$ is the estimated membership probability vector, $\gamma(e)$ is the true membership probability vector for data point a, and $|a.b| = a^T b$ for two vectors a and b.

To calculate the true membership probability vector, we create a smoothed probability vector by assuming a value of one for tags assigned to the document, and a value of 0.01 for tags that are not assigned to the document. We then normalize the vector to get mixed membership assignments for the document.

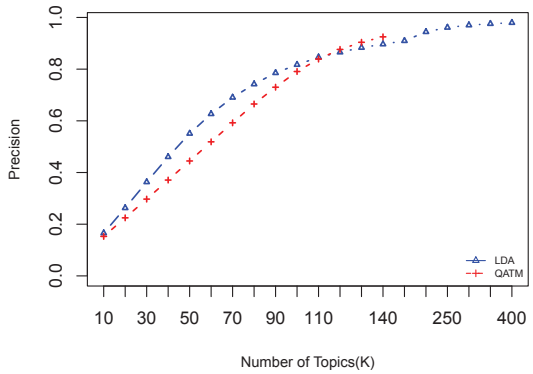
LDA configuration

For the Automatic Tagging task, LDA achieves better clustering performance with $\alpha = 100/k$. Figure 4.10 demonstrates this ($\beta = 0.01$ in both experiments.)

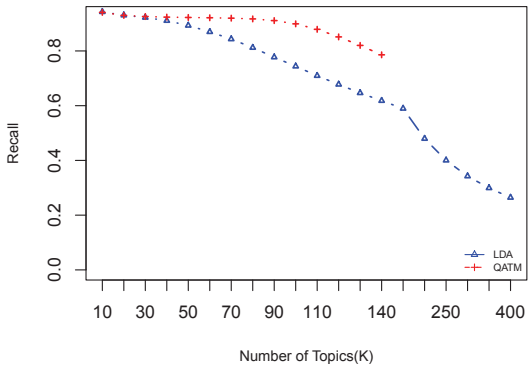
Results

Figure 4.11 compares precision, recall and F-measure for our method with values obtained for LDA. We compared the mixed membership probability vectors produced by our model and LDA with the tag distribution obtained from the data, using the metrics described in Eq. 4.10 and 4.11. It appears both methods achieve similar

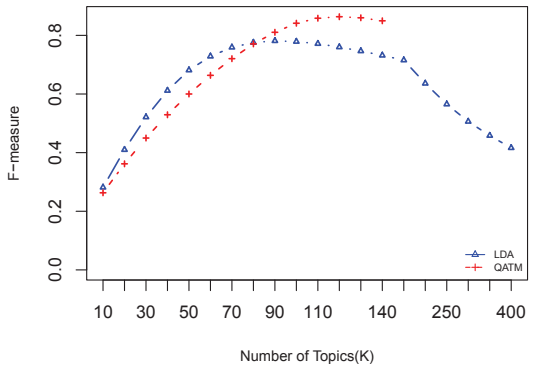
performance for small number of topics. However, for larger numbers of topics, our model achieves better clustering performance in terms of F-measure.



(a) Precision



(b) Recall



(c) F-measure

Figure 4.11: Clustering results.

Chapter 5

Conclusions

Community Question Answering services provide a rich source of practical information. With the growth of the web these services are employed more and more by users. To address their needs and utilize this vast source of knowledge, methods that exploit these archives are needed. This thesis presented a new probabilistic model, QATM, for representing the content of CQA archives. The model was designed to capture the archive's topical dependencies. In particular, three main assumptions were involved in its design:

1. Topics in answers are influenced by topics in questions.

2. Topics in answers are more technical and more specific than topics in questions.

3. Answers contain additional topics that are correlated with the topics in the question.

To analyze the model, it was applied for the tasks of Automatic Tagging, and Question Answering. In the Automatic Tagging task, QATM obtained almost similar performance as LDA, but for larger numbers of topics, it obtained better clustering. For Question Answering, we argued that traditional methods, such as TFIDF or Language Models, rely on word occurrence and are effective when the corpus contains lexically similar content with sufficient word overlap. Hence, because of the short length of questions and answers on CQAs and their sparse representations, we expected lexical based methods to perform worse than a topic-based method. However, the results showed the reverse. Further analysis of the experimental results in Section 4.3.1 provided insights about the dataset. The results denoted that the nature of our dataset was such that it favored word-based methods such as TFIDF. Particularly, the ground truth was created from accidental duplicates. As described by the website:

> ... their overlap (with another question) is not ambiguous; the question uses the same words and asks the same fundamental question, with no variation at all ...

Our analysis of the experimental results also confirmed the fact that these duplicates were more lexically similar than semantically similar to the data in our train data. In addition, the fact that our dataset was chosen from a technical CQA website, where the vocabulary domain is restricted and users are more likely to use the same recurring words, may also have contributed to the final results.

For future work, there remain a number of modifications and extensions that can be made to both the dataset and the model itself:

1. The ground-truth dataset was constructed using accidental duplicates from Stack Overflow. Alternatively, it could have been constructed using the Borderline duplicates on Stack Overflow. The questions in this set cover the same grounds as a previous question in the archive but their overlap with that question is ambiguous. Consequently, the uniqueness of these questions is subject to interpretation. These duplicates are tagged by users of Stack Overflow so that they naturally group with other relevant questions. A possible extension to the current work is to consider constructing a ground truth from this set.

2. The content of programming forums contains a combination of text and code. These forums naturally fall within the category of technical writings (communication) where the language is clear, unambiguous, concise and efficient in terms of chosen vocabulary. As a result, the domain of their vocabulary is restricted, and their content contains frequently occurring domain-specific words. In contrast, general purpose forums such as Yahoo! answers, utilize a more extensive vocabulary; their language is lucid and their structure more flexible. A possible extension to the current work is to evaluate the performance of the model against a dataset extracted from these general purpose forums and to test whether our assumptions, with which the model was designed, apply to these forums.

3. Topic-based models provide coarse topics inferred using the whole corpus [13].

Using only these broad topics to represent documents, can hurt retrieval performance. To resolve this over-generalization issue, a common approach is to use an ad-hoc method in which a lexical-based representation is combined with topic-based representations [30, 31]. Future work can also consider improving the performance of QATM by combining its output with a lexical-based representation such as Language models or TFIDF.

Bibliography

[1] Eugene Agichtein, Carlos Castillo, Debora Donato, Aristides Gionis, and Gilad Mishne. Finding high-quality content in social media. In *WSDM '08: Proceedings of the International Conference on Web Search and Web Data Mining*, pages 183–194, New York, NY, USA, 2008. ACM.

[2] Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Inf. Retr.*, 12:461–486, August 2009.

[3] Amit Bagga and Breck Baldwin. Entity-based cross-document coreferencing using the vector space model. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ACL '98, pages 79–85, Stroudsburg, PA, USA, 1998. Association for Computational Linguistics.

[4] Jiang Bian, Yandong Liu, Eugene Agichtein, and Hongyuan Zha. A few bad votes too many?: towards robust ranking in social media. In *Proceedings of the 4th International Workshop on Adversarial Information Retrieval on the Web*, AIRWeb '08, pages 53–60, New York, NY, USA, 2008. ACM.

[5] Jiang Bian, Yandong Liu, Eugene Agichtein, and Hongyuan Zha. Finding the right facts in the crowd: factoid question answering over social media. In Jinpeng Huai, Robin Chen, Hsiao-Wuen Hon, Yunhao Liu, Wei-Ying Ma, Andrew Tomkins, and Xiaodong Zhang, editors, *WWW*, pages 467–476. ACM, 2008.

[6] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[7] Robin D. Burke, Kristian J. Hammond, Vladimir A. Kulyukin, Steven L. Lytinen, N. Tomuro, and S. Schoenberg. Question answering from frequently asked question files: Experiences with the FAQ finder system. Technical report, Chicago, IL, USA, 1997.

[8] Asli Celikyilmaz, Dilek Hakkani-Tur, and Gokhan Tur. LDA based similarity modeling for question answering. In *Proceedings of the NAACL HLT 2010 Workshop on Semantic Search*, SS '10, pages 1–9, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[9] Chaitanya Chemudugunta, Padhraic Smyth, and Mark Steyvers. Modeling general and specific aspects of documents with a probabilistic topic model. In Bernhard SchÃlkopf, John Platt, and Thomas Hoffman, editors, *NIPS*, pages 241–248. MIT Press, 2006.

[10] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by Latent Semantic Analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 41(6):391–407, 1990.

[11] Christiane Fellbaum, editor. *WordNet: an electronic lexical database*. MIT Press, 1998.

[12] Scott A. Golder and Bernardo A. Huberman. Usage patterns of collaborative tagging systems. *J. Inf. Sci.*, 32:198–208, April 2006.

[13] Gregor Heinrich. Parameter estimation for text analysis. *Web: http://www. arbylon. net/publications/text-est.pdf*, 2005.

[14] Gregor Heinrich, Jörg Kindermann, Codrina Lauth, Gerhard Paaß, and Javier Sanchez-Monzon. Investigating word correlation at different scopes—a latent concept approach. In *Workshop Lexical Ontology Learning at Int. Conf. Mach. Learning*, 2005.

[15] Thomas Hofmann. Probabilistic Latent Semantic Indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 50–57, New York, NY, USA, 1999. ACM.

[16] Jiwoon Jeon, Bruce W. Croft, Joon H. Lee, and Soyeon Park. A framework to predict the quality of answers with non-textual features. In *SIGIR '06: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 228–235, New York, NY, USA, 2006. ACM Press.

[17] Jiwoon Jeon, W. Bruce Croft, and Joon Ho Lee. Finding similar questions in large question and answer archives. In Otthein Herzog, Hans-Jrg Schek, Norbert Fuhr, Abdur Chowdhury, and Wilfried Teiken, editors, *CIKM*, pages 84–90. ACM, 2005.

[18] Jeongwoo Ko, Luo Si, and Eric Nyberg. A probabilistic framework for answer selection in question answering. In *In Proceedings of NAACL/HLT*, 2007.

[19] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 1951.

[20] Mingrong Liu, Yicen Liu, and Qing Yang. Predicting best answerers for new questions in community question answering. In *Proceedings of the 11th International Conference on Web-age Information Management*, WAIM'10, pages 127–138, Berlin, Heidelberg, 2010. Springer-Verlag.

[21] Andrew Kachites McCallum. MALLET: A Machine Learning for Language Toolkit. *http://mallet.cs.umass.edu*, 2002.

[22] Yajie Miao, Lili Zhao, Chunping Li, and Jie Tang. Automatically grouping questions in yahoo! answers. In *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '10, pages 350–357, Washington, DC, USA, 2010. IEEE Computer Society.

[23] David Mimno and Andrew McCallum. Expertise modeling for matching papers with reviewers. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 500–509, New York, NY, USA, 2007. ACM.

[24] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 275–281, New York, NY, USA, 1998. ACM.

[25] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18:613–620, November 1975.

[26] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. In *INFORMATION PROCESSING AND MANAGEMENT*, pages 513–523, 1988.

[27] Mahdi Shafiei and Hugh Chipman. Mixed-membership stochastic block-models for transactional networks. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 1019–1024, Washington, DC, USA, 2010. IEEE Computer Society.

[28] Ellen M. Voorhees. The TREC-8 question answering track report. In *Proceedings of TREC-8*, pages 77–82, 1999.

[29] Kai Wang, Zhaoyan Ming, and Tat-Seng Chua. A syntactic tree matching approach to finding similar questions in community-based QA services. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 187–194, New York, NY, USA, 2009. ACM.

[30] Xing Wei and W. Bruce Croft. LDA-based document models for ad-hoc retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 178–185, New York, NY, USA, 2006. ACM.

[31] Xing Yi and James Allan. A comparative study of utilizing topic models for information retrieval. In *Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval*, ECIR '09, pages 29–41, Berlin, Heidelberg, 2009. Springer-Verlag.

Appendix A

Derivations

Factoring the joint distribution in Equation 3.2:

$$
\begin{aligned}
&\mathrm{P}(W_Q, W_A, Z_Q, Y_A, Z_A|\alpha_\theta, \alpha_{\beta_Q}, \alpha_{\beta_A}, \alpha_\phi) \; = \\
&\iiiint \mathrm{P}(W_Q, W_A, Z_Q, Y_A, Z_A, \theta_Q, \phi, \beta_Q, \beta_A|\alpha_\theta, \alpha_{\beta_Q}, \alpha_{\beta_A}, \alpha_\phi)\mathrm{d}\theta_Q\mathrm{d}\phi\mathrm{d}\beta_Q\mathrm{d}\beta_A \; = \\
&\int \mathrm{P}(W_Q|Z_Q, \beta_Q)\mathrm{P}(\beta_Q|\alpha_{\beta_Q})\mathrm{d}\beta_Q \times \qquad \text{(A.1)} \\
&\int \mathrm{P}(W_A|Z_A, \beta_A)\mathrm{P}(\beta_A|\alpha_{\beta_A})\mathrm{d}\beta_A \times \qquad \text{(A.2)} \\
&\int \mathrm{P}(Z_Q|\theta_Q)\mathrm{P}(Y_A|\theta_Q)\mathrm{P}(\theta|\alpha_\theta)\mathrm{d}\theta_Q \times \qquad \text{(A.3)} \\
&\int \mathrm{P}(Z_A|Y_A, \phi)\mathrm{P}(\phi|\alpha_\phi)\mathrm{d}\phi \qquad \text{(A.4)}
\end{aligned}
$$

The conditional A.1 can be rewritten into two products:

$$
\begin{aligned}
&\int \mathrm{P}(W_Q|Z_Q, \beta_Q)\mathrm{P}(\beta_Q|\alpha_{\beta_Q})\mathrm{d}\beta_Q = \\
&\int \prod_{i=1}^{N_Q}\prod_{n=1}^{N_{Q_i}} \mathrm{P}(W_{Q_i}^n|Z_{Q_i}^n, \beta_Q)\mathrm{P}(\beta_Q|\alpha_{\beta_Q})\mathrm{d}\beta_Q = \\
&\int \prod_{k=1}^{K}\prod_{v=1}^{V} \left[\beta_Q^{kv}\right]^{C_Q^{kv}} \prod_{k=1}^{K}\left(\frac{\Gamma(V_Q\alpha_{\beta_Q})}{\Gamma(\alpha_{\beta_Q})^{V_Q}}\prod_{v=1}^{V_Q}\left[\beta_Q^{kv}\right]^{\alpha_{\beta_Q}-1}\right)\mathrm{d}\beta_Q = \\
&\prod_{k=1}^{K}\frac{\Gamma(V_Q\alpha_{\beta_Q})}{\Gamma(\alpha_{\beta_Q})^{V_Q}}\frac{\prod_{v=1}^{V_Q}\Gamma(\alpha_{\beta_Q}+C_Q^{kv})}{\Gamma\left[\sum_{v=1}^{V_Q}(\alpha_{\beta_Q}+C_Q^{kv})\right]} \qquad \text{(A.5)}
\end{aligned}
$$

where C_Q^{kv} is the number of times word v is assigned to Q-topic k. The conditional A.2 can be also be rewritten into two products:

$$\int \mathrm{P}(W_A|Z_A,\beta_A)\mathrm{P}(\beta_A|\alpha_{\beta_A})\mathrm{d}\beta_A = \prod_{l=1}^{L}\frac{\Gamma(V_A\alpha_{\beta_A})}{\Gamma(\alpha_{\beta_A})^{V_A}}\frac{\prod_{v=1}^{V_A}\Gamma(\alpha_{\beta_A}+C_A^{lv})}{\Gamma\left[\sum_{v=1}^{V_A}(\alpha_{\beta_A}+C_A^{lv})\right]} \tag{A.6}$$

where C_A^{lv} is the number of times word v is assigned to A-topic l. Similarly, the conditional A.3 can be rewritten as:

$$\begin{aligned}
&\int \mathrm{P}(Z_Q|\theta_Q)\mathrm{P}(Y_A|\theta_Q)\mathrm{P}(\theta|\alpha_\theta)\mathrm{d}\theta_Q = \\
&\int \prod_{i=1}^{N_Q}\prod_{n=1}^{N_{Q_i}}\mathrm{P}(Z_{Q_i}^n|\theta_{Q_i})\prod_{i=1}^{N_Q}\prod_{j=1}^{N_{A_{Q_i}}}\prod_{n=1}^{N_{A_j}}\mathrm{P}(Y_{A_{i,j}}^n|\theta_{Q_i})\mathrm{P}(\theta_Q|\alpha_\theta)\mathrm{d}\theta_Q = \\
&\int \prod_{i=1}^{N_Q}\prod_{k=1}^{K}\left[\theta_{Q_i}^k\right]^{C_{Q_i}^k}\prod_{i=1}^{N_Q}\prod_{k=1}^{K}\left[\theta_{Q_i}^k\right]^{C_{A_{Q_i}}^k}\prod_{i=1}^{N_Q}\frac{\Gamma(k\alpha_\theta)}{\Gamma(\alpha_\theta)^k}\prod_{k=1}^{K}\left[\theta_{Q_i}^k\right]^{\alpha_\theta-1} = \\
&\prod_{i=1}^{N_Q}\frac{\Gamma(K\alpha_\theta)}{\Gamma(\alpha_\theta)^K}\frac{\prod_{k=1}^{K}\Gamma(\alpha_\theta+C_{Q_i}^k+C_{A_{Q_i}}^k)}{\Gamma\left[\sum_{i=1}^{K}(\alpha_\theta+C_{Q_i}^i+C_{A_{Q_i}}^i)\right]}
\end{aligned} \tag{A.7}$$

where $C_{Q_i}^k$ is the number of times Q-topic k is assigned to words in question Q_i and $C_{A_{Q_i}}^k$ denotes the number of times A-topics for words in the set of answers for question Q_i are drawn conditioned on Q-topic k. Similarly the conditional A.4 can be rewritten as:

$$\begin{aligned}
&\int \mathrm{P}(Z_A|Y_A,\phi)\mathrm{P}(\phi|\alpha_\phi)\mathrm{d}\phi = \\
&\int \prod_{i=1}^{N_Q}\prod_{j=1}^{N_{A_{Q_i}}}\prod_{n=1}^{N_{A_{i,j}}}\mathrm{P}(Z_{A_{i,j}}^n|Y_{A_{i,j}},\phi)\prod_{k=1}^{K}\mathrm{P}(\phi_k|\alpha_\phi)\mathrm{d}\phi = \\
&\int \prod_{k=1}^{K}\prod_{l=1}^{L}[\phi_{kl}]^{C_k^l}\prod_{k=1}^{K}\frac{\Gamma(L\alpha_\phi)}{\Gamma(\alpha_\phi)^l}\prod_{l=1}^{L}[\phi_{kl}]^{\alpha_\phi-1}\,\mathrm{d}\phi = \\
&\prod_{k=1}^{K}\frac{\Gamma(L\alpha_\phi)}{\Gamma(\alpha_\phi)^L}\frac{\prod_{l=1}^{L}\Gamma(C_k^l+\alpha_\phi)}{\Gamma\left[\sum_{i=1}^{L}(\alpha_\phi+C_k^l)\right]}
\end{aligned} \tag{A.8}$$

where C_k^l is the number of times an A-topic l is drawn conditioned on a Q-topic k in the entire corpus.

Using the above equations the update equation from which the Gibbs sampler draws the Q-topic for a question word is:

$$\begin{aligned}
&\mathrm{P}(Z_{Q_i}^n = k | Z_Q^{-n}, Y_A, Z_A, W_Q, W_A) \propto \\
&\frac{\alpha_{\beta_Q} + C_Q^{kW_{Q_i}^n}}{\sum_{v=1}^{V}(\alpha_{\beta_Q} + C_Q^{kv})} \frac{\alpha_\theta + C_{Q_i}^k + C_{A_{Q_i}}^k}{\sum_{i'=1}^{k}(\alpha_\theta + C_{Q_i}^{i'} + C_{A_{Q_i}}^{i'})} \\
&\text{update } C_Q^{kW_{Q_i}^n}, C_{Q_i}^k \text{ and decrement count by one}
\end{aligned} \tag{A.9}$$

Similarly, the update equation for drawing the A-topic and Q-topic of a word in the answer is:

$$\begin{aligned}
&\mathrm{P}(Y_{A_{i,j}}^n = k, Z_{A_{i,j}}^n = l | Y_{A_{i,j}}^{-n}, Z_{A_{i,j}}^{-n}, Z_Q, W_Q, W_A) \propto \\
&\frac{\alpha_{\beta_A} + C_A^{lW_{A_{i,j}}^n}}{\sum_{v=1}^{V}(\alpha_{\beta_A} + C_A^{lv})} \frac{\alpha_\theta + C_{Q_i}^k + C_{A_{Q_i}}^k}{\sum_{i'=1}^{k}(\alpha_\theta + C_{Q_i}^{i'} + C_{A_{Q_i}}^{i'})} \frac{C_k^l + \alpha_\phi}{\sum_{i=1}^{L}(\alpha_\phi + C_k^i)} \\
&\text{update } C_k^l, C_A^{lW_{A_{i,j}}^n}, C_{A_{Q_i}}^k \text{ and decrement count by one}
\end{aligned} \tag{A.10}$$

Appendix B

Main Entities on Stack Overflow

1. **Questions** Software questions posted by users

   Properties include

   (a) Tags

   (b) Author

   (c) Total points

   (d) Votes

   (e) Favorite

   (f) Can be commented on, edited, revised

2. **Answers** Answers posted to questions by users

   Properties include

   (a) Author

   (b) Point

   (c) Votes

   (d) Can be commented on, edited, revised

3. **Users** Members of the community who post or answer questions .

   Properties include

   (a) Profile

   (b) Reputation, profile views, badges, etc

   (c) Questions asked (Question ID, how many times the question has been favorite, number of answers, number of views, tags, person who answered and his reputation)

   (d) Questions answered(Question ID, total number of votes for the answers)

(e) Total number of votes (positive and negative)

(f) Total number of tags

(g) Total number of badges

(h) Detailed overview of activity(badges awarded, comments, answers, revisions, etc)

(i) Reputation

(j) Favorites

(k) Accounts

Appendix C

Dataset Extraction

C.1 Train Data

In brief, the Train Data Preparation Steps:

1. We collected statistics about tag frequencies and also tag co-occurrence frequencies. To be specific, the tag frequency of questions with more than two answers was calculated. In addition, the pairwise tag frequency of questions with more than two answers was calculated.

2. These statistics were examined, and with regard to our objectives we manually selected a total of 21 tags. We chose 7 tags that were popular and that co-occurred frequently with other tags, around 7 tags that were popular but co-occurred less frequently, around 7 tags that were popular and rarely co-occurred with other tags. The tags in Table 4.2 were chosen.

3. For each of the tags selected in the previous step, we randomly collected 200 questions. 4200 questions were extracted.

4. For each of the questions selected in the previous step, we extracted at most four of its most relevant answers. At the end of this step, we had extracted 15,822 answers.

C.2 Test Data (Duplicate Questions)

To compare the answer retrieval performance of our model with existing methods, we extracted a ground truth set from the accidental duplicates. To extract the duplicates the following steps were taken:

1. Duplicate questions that had been voted on and closed as exact duplicates were identified and extracted. 5783 questions were marked as duplicates in the January data dump.

2. This group was refined to extract questions that were duplicates of the questions in our train dataset. 822 duplicate questions were found. Since each duplicate question may be a duplicate of multiple questions and a question may have multiple duplicates (cardinality of the relation is many to many) these 822 duplicate questions correspond to 852 relations.

3. The 822 duplicates were split into two subsets consisting of 700 duplicate questions for the train set and 122 duplicate questions for the validation set. The 700 duplicate questions correspond to 722 duplication relations and the 122 duplicates in the validation set correspond to 130 duplication relations.

C.3 Examples of Duplicate Questions on Stack Overflow

1. Original Question (Train 113547)

   URL `http://stackoverflow.com/questions/113547/`

   Title iPhone development on Windows

   Content Is there a way to develop iPhone (iOS) applications on Windows? I really don't want to get yet another machine. There is a project on http://code.google.com/p/winchain/wiki/HowToUse that seemed to work with iPhone 1.0, but had limited success with iPhone 2.0, plus it requires all the Cygwin insanity.Is there anything else, or do I have to buy a Mac?

2. Accidental Duplicate (Test 68196), lexically similar to original question

   URL `http://stackoverflow.com/questions/68196/`

   Title Develop iPhone applications using Microsoft Windows [closed]

   Content I don't have a mac, but I wish to develop iPhone applications on Windows platform. Is this possible?

3. Borderline Duplicate, semantically related to the original question

URL `http://stackoverflow.com/questions/22358/`

Title How can I develop for iPhone using a Windows development machine?

Content Is there any way to tinker with the iPhone SDK on a Windows machine? Are there plans for an iPhone SDK version for Windows? The only other way I can think of doing this is to run a Mac VM image on a VMWare server running on Windows, although I'm not too sure how legal this is.

4. Borderline Duplicate, semantically related to the original question

   URL `http://stackoverflow.com/questions/2642877/`

   Title Best windows iphone app development alternative

   Content What do you think is the best way to develop iphone apps on windows? What are the pros / cons of your method, and why do you use it over other options? How complex is your method in relation to other options? I am more interested in standalone and web apps but fell free to discuss gaming graphics. Yes I know you need to build on a mac to be able to put it on the app store, so no "use a mac" answers please.

5. Borderline Duplicate, semantically related to the original question

   URL `http://stackoverflow.com/questions/2261267/`

   Title iPhone development on PC

   Content Can anybody shortly describe solutions to start develop for iPhone on PC?

6. Borderline Duplicate, semantically related to the original question

   URL `http://stackoverflow.com/questions/2438718/`

   Title developing iphone apps on windows is it worth the hassel

   Content I'm only after a simple solution and won't be developing anything particularly complex. But I'm wondering whether the hassals of developing an iPhone app NOT on MacOS are really that significant to avoid giving it a shot. Bearing in mind that I do have access to a mac every now and

again. So I would be able to compile it using the official Apple supported SDK, but I just want to be able to develop it in my own environment (windows laptop). I heard someone mention a while ago that there are various objective C compilers that allow writing code in various other web technologies as well. Are these really valid. And am I alone in thinking Apple's whole attitude towards this is totally imoral. Charging 200 for the privelege of having your app unequivocally rejected etc etc and then not being allowed to look directly at Steve Jobs or his golden retrievers.

Appendix D

Question-All-Answer Retrieval

To evaluate the answer retrieval performance of our model, we designed two types of retrieval tasks. In the first task, referred to as *Question-Answer-Pair Retrieval*, the model was used to retrieve and rank documents consisting of a question and answer pair. We reported the experimental results for this type of retrieval in Section 4.3.1. For the second task, referred to as *Question-All-Answers Retrieval*, we combined each question with at most four of its relevant answers into one document. Hence, each document in the train set consisted of a question and its several answers. We then used different models (QATM,TFIDF, and LDA) to retrieve and rank those documents.

D.1 LDA Configuration

To find the optimal value of LDA hyperparameter α, we ran two sets of experiments with $\alpha = 50/k$ and $\alpha = 100/k$ ($\beta = 0.01$ in both experiments). The former resulted in better performance as shown in Figure D.1.

D.2 Results

Figure D.2 denotes MAP performance results for the Question-All-Answers-Retrieval experiments. As the number of topics increase, LDA performs better compared to our model. Table D.1 compares the TopN performance results. Similar to the Question-Answer-Pair retrieval experiments, TFIDF outperforms both of the topic-based approaches.

D.3 Analysis of Results

Figure D.3 shows the rank distribution for QATM and TFIDF, where the rank domain has been broken into intervals of 100. From this figure it can be seen that the retrieval results of TFIDF are superior to those of QATM. Similar to Section 4.3.1,

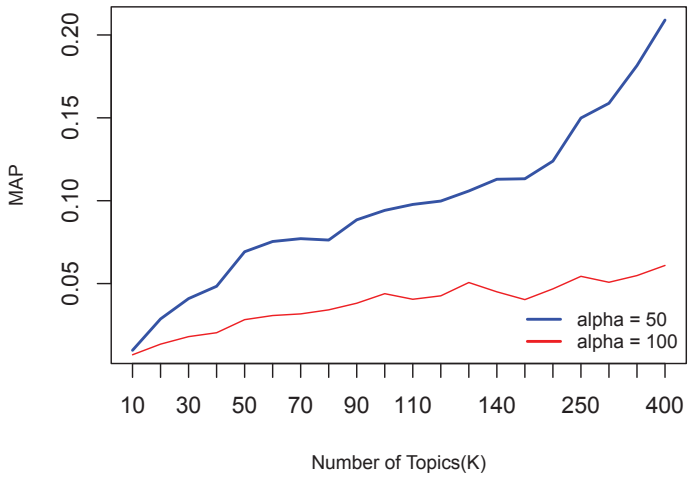


Figure D.1: LDA configuration.

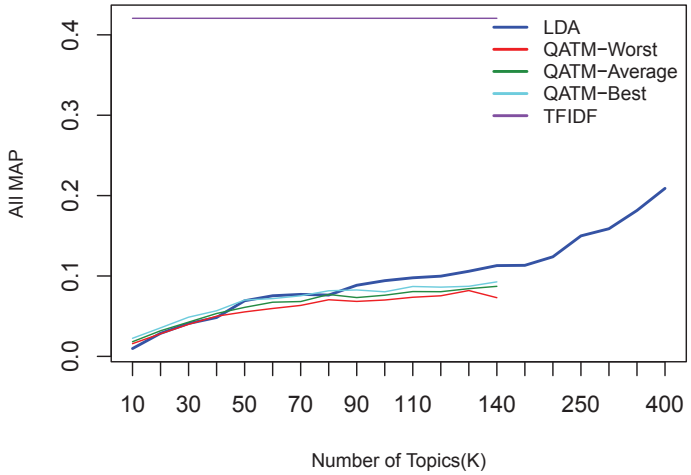
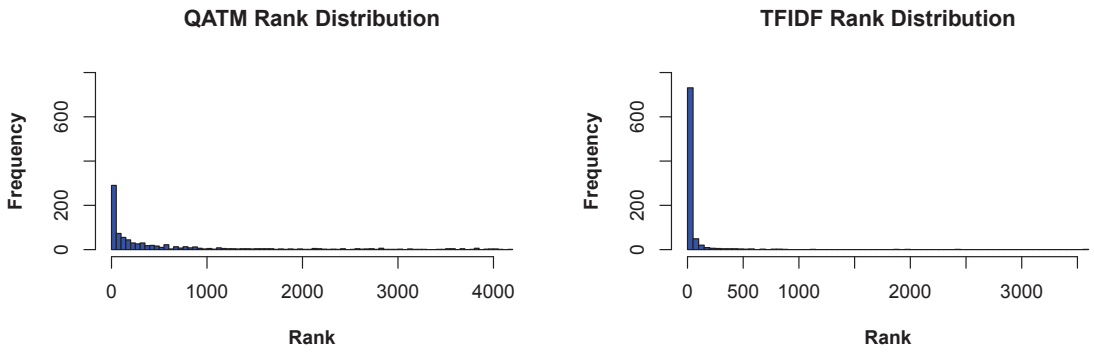


Figure D.2: MAP results for question-all-answers retrieval.

	Top1	Top2	Top3	Top4	Top5
LDA	0.054	0.074	0.083	0.088	0.091
QATM-Worst	0.041	0.05	0.054	0.056	0.058
QATM-Average	0.049	0.06	0.066	0.069	0.071
QATM-Best	0.052	0.066	0.072	0.074	0.076
TFIDF	0.304	0.353	0.372	0.383	0.392

Table D.1: TopN results for question-all-answers retrieval



(a) QATM rank distribution

(b) TFIDF rank distribution

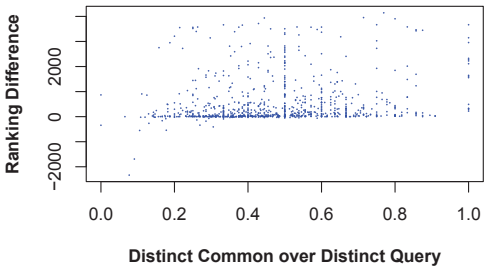
Figure D.3: QATM and TFIDF rank distribution.

the data is examined using metrics that describe the lexical overlap between queries and documents (i.e in Question-All-Answer retrieval each document consists of a question and at most four of its relevant answers).

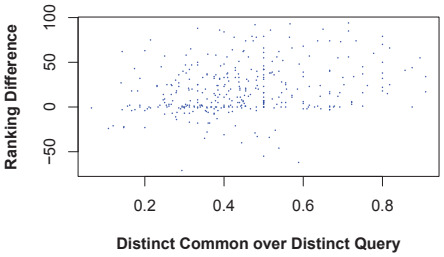
The 822 duplicates(queries) in our ground truth correspond to 852 relevant documents in the train data. Hence, for each of these data points, we calculated the measures presented in Section 4.3.1. Figure D.4 plots the difference in ranks produced by QATM and TFIDF against the DCoDQ of each data point. Figure D.5 plots the difference in ranks against the LCoLQ.

For these 852 data points, we also calculated the percentage of documents that are retrieved in the top 100 results for their corresponding query. This percentage is 42% for QATM, 91% for TFIDF and an overlap of 41%. Figures D.4b and D.5b denote the ranking difference against lexical similarity of these data items.

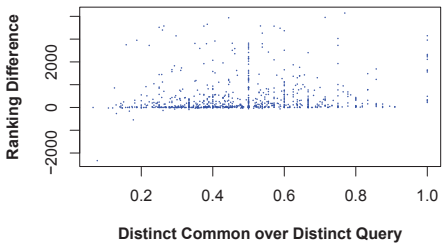
The figures confirm the conclusion in Section 4.3.1; that the ground truth and train data are more lexially similar than semantically similar.



(a) All datapoints

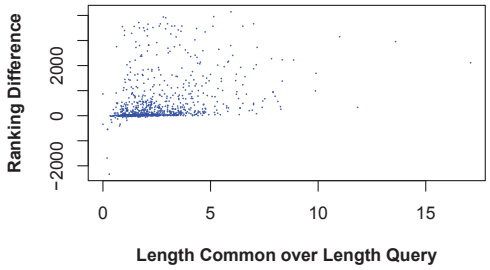


(b) Datapoints for which both methods produce rank less than 100

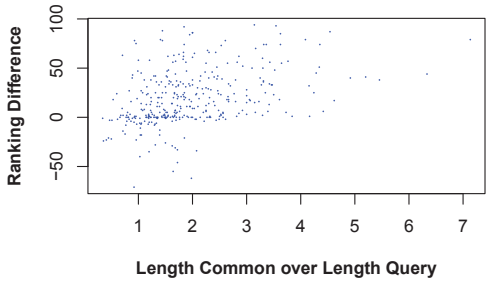


(c) Datapoints with at least one method producing rank less than 100

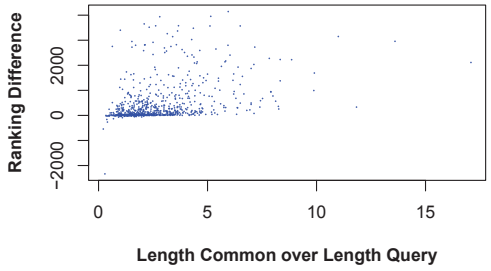
Figure D.4: Ranking difference of datapoints against their lexical overlap (DCoDQ). Data points in the left of the diagrams correspond to query questions that do not have much word overlap with the retrieved question-answer pair.



(a) All datapoints



(b) Datapoints for which both methods produce rank less than 100



(c) Datapoints with at least one method producing rank less than 100

Figure D.5: Ranking difference of datapoints against their lexical overlap(LCoLQ). Data points in the left of the diagrams correspond to query questions that do not have much word overlap with the retrieved question-answer pair.

Appendix E

Computational Costs

The computational cost of inferring the parameters of any probability distribution is directly proportional to the dimensions (the number of parameters) of the distribution. For topic models, inferring their parameters can be a relatively difficult problem if they specify a high-dimensional probability distribution. The complexity of LDA is compared to K-means in [30] and it is concluded that the two algorithms are similar in terms of efficiency and running time.

The Gibbs sampling procedure in LDA is linear with $I, K, M * \bar{W}_D$ where I is the number of iterations, K is the number of topics, M is the number of documents, and $\bar{W}_D$ is the average number of words in one document.

However, our model has more parameters compared to LDA; QATM has two sets of topics, Q-topics and A-topics. It is linear with $I, K, L, M * (\bar{W}_Q + 2 * \bar{W}_A)$ where I is the number of iterations, K is the number of Q-topics, L is the number of A-topics, M is the number of documents, $\bar{W}_Q$ is the average number of tokens in a question, $\bar{W}_A$ is the average number of tokens in the answers of a question, and $\bar{W}_Q + \bar{W}_A = \bar{W}_D$.

In terms of space requirements, the Gibbs sampling procedure for LDA uses mainly five large data structures [13]: there are two count variables, which are matrices with dimensions $M * K$ and $K * V$ where M is the number of documents, K is the number of topics, and V is the length of the vocabulary of the corpus. In addition, the row sums of these count variables are also stored, corresponding to a dimension of M and K respectively. The fifth data structure is the state variable Z_m^n which is stored for every word in every document, resulting in a dimension of $M * \bar{W}_D$.

QATM requires the count variable C_Q^{KV}, which is a matrix with a dimension of $K * V$, its row sum C^K which has a dimensionality of K, the count variable C_A^{LV} which is matrix with dimensionality of $L * V$, its row sum C^L which has a dimensionality of L. The count variable C^{QK} with a dimensionality of $Q * K$, where Q is the number of questions in the corpus (it is equal to the number of documents M). the count variable

C_A^{QK} with a dimensionality of $Q * K$, the count variable C^{LK} with a dimensionality of $L * K$. The state variable $Z_{Q_i}^n$ is stored for every word in the questions in the corpus, resulting in a dimension of $Q * \bar{W}_Q$. The state variables $Z_{A_{i,j}^n}$, $Y_{A_{i,j}^n}$ are stored for every word in the answers of a question in the corpus, resulting in a dimension of $2 * Q * \bar{W}_A$.

The training and test phases for both QATM and LDA were run on ACEnet [1]. ACEnet has a number of clusters. We ran experiments on the Placentia cluster [2]. This cluster has 114 nodes, with a total of 856 cores. The amount of ram per node is either 8 GB, 16 GB, or 32 GB. The CPU's range from Dual-Core AMD Opteron 2.6 GHz to Quad-Core AMD Opteron 2.7 GHz. Specific details are on the website.

The number of Gibbs iterations for QATM in the training phase was 3000, and it was 5000 in the test phase. For LDA, the number of Gibbs iterations for both the training and test phase was 2000.

[1] `http://www.ace-net.ca/wiki/ACEnet`
[2] `http://www.ace-net.ca/wiki/Placentia`