Virginia Commonwealth University
VCU Scholars Compass

Theses and Dissertations

Graduate School

2019

Applying Hierarchical Tag-Topic Models to Stack Overflow

John J. Coogle
VCU

APPLYING HIERARCHICAL TAG-TOPIC MODELS TO STACK OVERFLOW

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

by

JOHN COOGLE

Bachelor of Science in Computer Science from Virginia Commonwealth University, 2017

Proposal Director: Dr. Kostadin Damevski,

Assistant Professor, Department of Computer Science

Virginia Commonwealth University

Richmond, Virginia

February, 2019

Acknowledgements

I would like to thank Dr. Kostadin Damevski for providing the initial direction for this work, for his constant guidance throughout the work, and for his reviews and suggestions on this work. Without these, this work would not have been possible.

I would like to thank Dr. Hui Chen for his many and varied contributions to the research behind this work, for his great expertise, for sharing his many tools and technologies for working with the L2H model, and for providing a server with which to train the model. Without these, the quality of this work would have suffered greatly.

I would like to thank my family for without their support, my graduate studies would not have been possible.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Abstract

APPLYING HIERARCHICAL TAG-TOPIC MODELS TO STACK OVERFLOW

By John Coogle

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

Virginia Commonwealth University, 2019.

Director: Dr. Kostadin Damevski,
Assistant Professor, Department of Computer Science

Stack Overflow is a question and answer site for programming questions. It has become one of the most widely used resources for programmers, with many programmers accessing the site multiple times per day. A threat to the continued success of Stack Overflow is the ability to efficiently search the site. Existing research suggests that the inability to find certain questions results in unanswered questions, long delays in answering questions, or questions which are unable to be found by future visitors to the site. Further research suggests that questions with poor tag quality are particularly vulnerable to these issues.

In this thesis, two approaches are considered for improving tag quality and search efficiency: automatic tag recommendations for question authors, and organizing the existing set of tags in a hierarchy from general to specific for Stack Overflow readers. A hierarchical organization is proposed for it's ability to assist exploratory searches of the site.

L2H, a hierarchical tag topic model, is a particularly interesting solution to these approaches because it can address both approaches with the same model. L2H is evaluated in detail on several proposed evaluation criteria to gauge it's fitness for addressing

these search challenges on Stack Overflow.

CHAPTER 1

INTRODUCTION

Modern software development is heavily reliant on the internet. In nearly any programming task, there is some knowledge that the programmer does not have, but can be found somewhere on the internet. Stack Overflow is a question and answer site specifically for programming related questions and making the best answers immediately available for future viewers. In this sense, Stack Overflow can be viewed almost like the programming equivalent to a site such as Wikipedia, where a programmer can search for some question on Stack Overflow and find a community created answer that often includes code related to the problem [1].

In filling this role, visits to Stack Overflow have become a vital part of everyday programming [2]. However, the nature of Stack Overflow has posed a number of sustainability challenges. Notably, the volume of questions and answers on the site makes it difficult to search efficiently. The unique mixture of both text and code in Stack Overflow is a challenge for common search algorithms and Natural Language Processing (NLP) techniques. This is particularly problematic because if a question is hard to find then it is both difficult for future visitors to benefit from it and difficult for experts to contribute a meaningful answer. Thus, the ability to search the site efficiently is vital to its overall usability.

In its current state, one of Stack Overflow's most notable searching features is the tag. In Stack Overflow, a tag is any user created word-or-phrase added to a question. Up to 5 tags can be assigned to a given question, and questions that have a tag in common are considered to share some similarity [3]. For example, two different questions that both have the JAVA tag are understood to both somehow involve the Java programming language.

Tags are used in a variety of different ways on Stack Overflow. For example, they can be used explicitly to filter questions with some combination of tags [4]. This could allow an expert to easily find questions that are within their realm of expertise to answer. Tags can also be used to complement a traditional search by narrowing it to certain topics. Finally, tags can be used implicitly by a search algorithm to improve the quality of results [5].

While tags are useful for searching, a major limitation is that they are entirely human generated. If a question is tagged poorly, it becomes significantly more difficult to find [5]. Furthermore, there is currently no way to improve a question's tagging aside from a person manually reviewing it and providing quality tags for the question.

Another limitation is tags can have vastly different levels of specificity. For example, JAVA is a very general tag that could apply to any question involving the Java programming language. On the other hand, .HTPASSWD is a very specific tag, referring to one particular file used specifically by Apache HTTP Server, and could apply to a much narrower range of questions. This creates difficulty for someone tagging a question: If a very specific tag like .HTPASSWD is used, then the question may be overlooked by people only looking at related, but more general tags such as the APACHE tag which applies to anything related to Apache HTTP Server. On the other hand, using very general tags can make it more difficult for domain experts to find the question. Ideally, the question would be well tagged with a mix of relevant general and specific tags, but it is unlikely that an average Stack Overflow user will know all of the tags that are relevant to their question. In practice, this means questions often have an incomplete set of tags that do not fully represent the question [5].

The ideal solution would be total automatic tagging, where some algorithm can look at any new question and automatically assign all appropriate tags correctly. In practice, creating an algorithm that performs well enough to automatically tag posts with sufficient accuracy is a very difficult problem. A more practical solution would be using an algorithm or set of algorithms to assist human generated tagging. Specifically, there are two key issues that should be addressed to improve human tagging:

1. Use an algorithm to recommend a set of potentially relevant tags that a human can then manually select from. The primary role of the algorithm in this case is not to have perfect tagging precision by itself, but rather to make human taggers aware of other types tags that could be relevant to their question and ultimately lead to better tag selection by the human tagger.

2. Organize tags in terms of their generality and specificity. With such an organization, it could help a person manually find related tags that could be relevant to their question. This could be especially useful to those who want a quick answer to their question. They are more likely to seek out a mixture of both general and specific tags [2] which is directly addressed by such an organization.

One possible way to address both issues is with a hierarchical tag-topic model such as Label-to-Hierarchy, or L2H [6]. As input, L2H takes a set of text documents tagged with some labels as training data. After training on those documents, it can then take new text documents and predict which labels may apply to it. In theory, that could address the first issue of recommending tags for a human tagger to use. However, L2H has the additional benefit of creating a hierarchy of labels during training. The most general tags are the root of the hierarchy, and more specific tags within that general category are the children. In theory, that could address the second issue of organizing tags by generality and specificity. Given both of these potential benefits, it is interesting to explore L2H's feasibility as a possible solution to both of those issues. Furthermore, L2H was originally designed for multi-labeled documents with a potentially incomplete set of human generated labels [6], implying L2H should be particularly suited for the Stack Overflow dataset.

This thesis will examine the suitability of L2H for addressing these issues in Stack

Overflow. The focus shall be on the performance of L2H itself and its general suitability as a candidate solution for these tagging issues. This thesis is based on work which has been submitted in a paper for review [7]. The performance of L2H will be evaluated in a variety of manners.

1. Does L2H produce a hierarchy organizing tags from general to specific? Metrics will be defined to quantify specificity and generality. Those metrics will be applied to ensure that deeper levels of the hierarchy are more specific and shallower levels are more general.

2. How effective is the hierarchy in finding related tags from known tags? Metrics will be defined to quantify how diverse the branches of a hierarchy are. Ideally, sibling branches should show reasonable diversities, yet sufficient commonality. If both this property is satisfied and the hierarchy is indeed organized from general to specific, then the hierarchy can be considered to do an effective job of organizing related tags near potentially known tags.

3. Does L2H predict similar tags to human chosen tags on unseen questions? L2H will be used to predict tags for unseen Stack Overflow questions. The results will be compared with the human created tags to ensure the tagging is similar.

4. How effective is L2H at recommending tags for human tagging? Additional metrics will be defined to examine whether a tag that disagrees with human tagging is likely to be a plausible tag for the question. Producing more plausible tags will be considered to be making more effective recommendations.

5. How effective is L2H at predicting tag synonyms? The L2H model will be used to predict which tags are synonymous. The results will be compared with human identified tag synonyms as a basic similarity metric to compare L2H's understanding of tag meaning compared with human understanding of tag meaning.

Additionally, the results will be examined for how suitable L2H is for identifying new tag synonyms, as well as what impact the hierarchy has on synonym identification ability

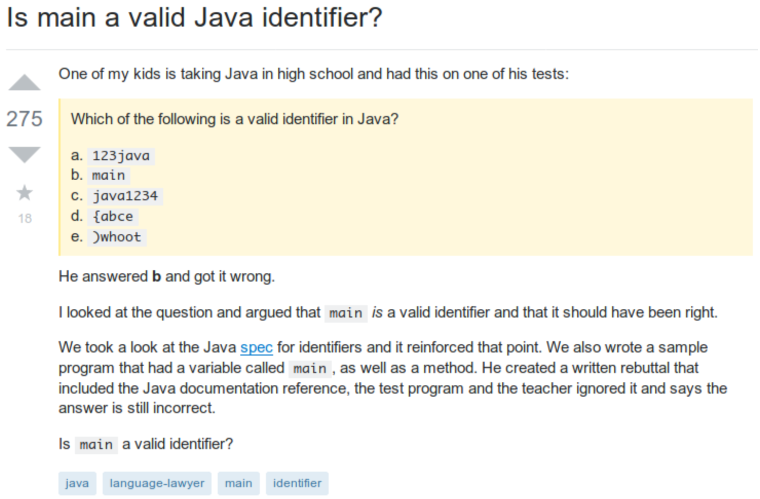
CHAPTER 2

BACKGROUND

On Stack Overflow, the strongest individual factor behind unanswered questions is question tagging [5]. In this chapter, the factors influencing tag quality are examined. From that, various models are examined for their suitability to improve both the quality and usage of tags with the ultimate goal of reducing the number of unanswered questions.

Stack Overflow is a community driven question and answer site specifically for programming and software engineering. Since its inception in 2008, Stack Overflow has grown to be one of the largest online programming resources in existence [8]. By the nature of the question and answer format, the site is only useful to the readers if the questions ultimately get answered. As Stack Overflow has gotten larger, an increasing number of questions have gone unanswered, [9] which poses a potential challenge to the long-term sustainability of the site.

Researchers have explored why questions on Stack Overflow are going unanswered [5]. In some cases, the question is not appropriate for the site in some way and therefore should go unanswered. However, it is much more common for a question to go unanswered because the asker does not pose the question effectively, such as not providing enough details in the question for someone to answer it. The most common common reason that a question goes unanswered is because it fails to attract an expert member who can answer it. Furthermore, if a question doesn't attract an expert, the most common culprit is incorrect tagging of the question [5]. Then a logical starting point to counteract the increasing number of unanswered question on Stack Overflow is to improve the quality of question tagging.



Is main a valid Java identifier?

One of my kids is taking Java in high school and had this on one of his tests:

> Which of the following is a valid identifier in Java?
>
> a. `123java`
> b. `main`
> c. `java1234`
> d. `{abce`
> e. `)whoot`

He answered **b** and got it wrong.

I looked at the question and argued that `main` *is* a valid identifier and that it should have been right.

We took a look at the Java spec for identifiers and it reinforced that point. We also wrote a sample program that had a variable called `main`, as well as a method. He created a written rebuttal that included the Java documentation reference, the test program and the teacher ignored it and says the answer is still incorrect.

Is `main` a valid identifier?

Figure 1: An example of an active Stack Overflow question that demonstrates a common successful tagging paradigm [12]

2.1 Question Tagging

Stack Overflow requires that a question have at least one tag, and allows a maximum of five tags. Most questions are somewhere in the middle, with 72 percent of questions having between 2 and 4 tags. Active questions on Stack Overflow tend to have a variety of tags that complement each other and relate to multiple aspects of the question. A common tagging paradigm for active questions is one or two tags for the most general and relevant technology for the question, and then possibly several more specific tags detailing more specific elements relevant to that question [2]. The question in Figure 1 is an active question that is a prime example of this paradigm. It is tagged with java, which is a very general tag for the language in question (1.4 million questions with the tag [3]). However, it is also tagged with the much more specific tags of language-lawyer (4 thousand questions [3]), main (2,400 questions [3]) and identifier (1 thousand questions [3]), which all bring emphasis to very specific, but relevant details for the question. Other active questions also often demonstrate this paradigm [10] [11].

In contrast, unanswered questions tend to have a variety of tags that do not necessarily add to each other or the question. For example, a common tagging paradigm among unanswered questions is to tag the question with each technology the asker is using even when those technologies aren't especially important to the question. It's also common for all tags in an unanswered question to be of a similar level of generality or specificity with not much mixture. Finally, unanswered questions are often missing some tags that are relevant to their question. An example of these types of questions will be examined in section 2.1.2.

2.1.1 Prominent Tags

While the notion of a general tag is intuitive, it is not a well defined concept. The concept of a prominent tag can be defined to better examine tagging behavior in practice. A prominent tag is any tag that is more common than some threshold. For example, one possible criteria is whether a tag is within the top N most common tags. Prominent tags are defined as such to capture the intuition that a general tag is likely to be common. Ideally, the threshold should be set such that an intuitively general tag is likely to also be prominent and vice-versa. A question can then have its tags classified as prominent or non-prominent as a well defined approximation to the intuitive notion of general and specific tags.

A reasonable threshold must be chosen for prominent tags to be a useful concept. As prominent tags are intended to approximate general tags, it is useful to look at the known properties of a general tag for guidance in selecting a threshold. Specifically, a general tag is known to be used on a large number of questions and is more likely to be named after a specific technology (especially a primary technology, such as IOS as opposed to a specific aspect of that technology, such as UITABLEVIEW, or specific versions, such as IOS-12). In contrast, specific tags tend to be used on fewer questions and are more likely to be named after concepts rather than technology. Therefore, a threshold can be selected by ensuring questions above the threshold are both sufficiently

Top X	**Primary Tech.**	**Secondary Tech.**	**Concepts**	**LCTP**	**PPTA**
10	10	0	0	4.32%	100%
25	23	2	1	1.63%	86.7%
50	44	2	4	0.787%	84.0%
100	77	6	17	0.428%	66.0%
200	138	13	49	0.231%	61.0%

Table 1.: Statistics for the top tags on Stack Overflow, with the Least Common Tag Prevalence (LCTP) and the Percentage Primary Technologies Added (PPTA) computed.

common and sufficiently likely to be named after particular technologies.

We define two novel metrics to help further quantify this decision. For a given set of tags, The Least Common Tag Prevalence (LCTP) is defined as the percentage of all Stack Overflow questions that the least common tag within the set applies to. For example, among the top 10 tags, the least common tag is IOS, which is currently used on 582,226 questions. As there are currently 13,472,769 questions on Stack Overflow, the LCTP would then be $\frac{582{,}226}{13{,}472{,}769} = 4.32\%$. Using this metric gives a sense of how common tags are for a given threshold.

Additionally, let $PT(X)$ be the number of tags in set X which are primary technologies, and let $TT(X)$ be the total number of tags in set X. For two given sets of tags, A and B, where B is a superset of A, the Percentage Primary Technologies Added (PPTA) is defined $\frac{PT(B)-PT(A)}{TT(B)-TT(A)}$. This metric equates to what percentage of new tags in set B are primary technology tags. Broadly, the higher the percentage in this metric, the greater the percentage of purely general tags. Therefore, PPTA gives a sense of the other criteria, that general tags tend to be named after specific technologies.

In Table 1, the results of several thresholds are computed. Each tag is counted as either a primary technology, such as JAVA or WINDOWS, a secondary technology, such as UITABLEVIEW or .HTACCESS which are components of larger primary technologies, or a concept, such as CLASS or SORTING. Additional, both LCTP and PPTA are computed for each threshold. PPTA is computed relative to the threshold above. For example, the PPTA for the top 50 is relative to the top 25. For the top 10, PPTA is computed against the empty set.

LCTP observes a power law decay with increasing tags. PPTA observes a more irregular decay pattern with increasing tags. The drop between the top 10 and top 25 is expected, as the PPTA for the top 10 is compared against the null set. In contrast, the drop between top 50 and top 100 is more significant because there is no a priori reason to expect it.

Given this data, the threshold for a prominent tag is chosen as any tag within the top 50 most common tags. Going beyond the top 50 has notable drawbacks on both criteria. The drop off in PPTA beyond the top 50 means that a notable increase in undesired tags are also included. By itself, that's not necessarily undesirable as primary technologies are still a strict majority of added tags. If going by PPTA alone, it might be reasonable to accept this error and continue increasing the threshold until PPTA is below 50%.

The other criteria, LCTP, suggests that increasing the threshold to that point may not be reasonable. By definition, any tags added beyond the top 50 are used on fewer than 0.787%, or 1 out of every 127, Stack Overflow questions. Although no strict boundary for "uncommon" has been defined, such scarcity is nevertheless stretching the intuition of a general tag being common.

An alternative data driven perspective is comparing to the LCTP of the top 10 tags, since the top 10 most frequent from a set of over 46,000 can reasonably be accepted as common. Viewed like this, any tags added beyond the top 50 must be more than $\frac{4.32}{0.787} \approx 5.49$ times as rare as the least common of the top 10 tags. Note this factor is relative to the *least* common of the top 10, not the mean, median, or other metric for centeredness. In other words, all such metrics would create a *larger* factor than this. Beyond the top 100, tags must be more rare than a factor of 10.1, and beyond the top 200, more than 18.7.

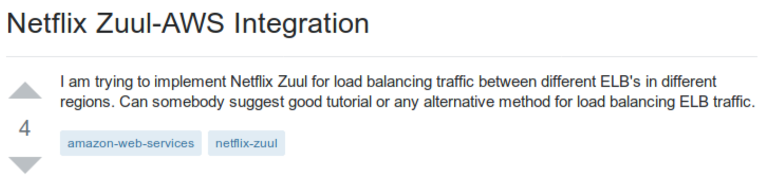
Beyond the top 100, all tags added are already more than an order of magnitude less common than any of the top 10 tags. That can be reasonably accepted as no longer common. Between the top 50 and top 100, the added tags are between 5.49 times and 10.1 times as rare as the least common of the top 10 tags. Although not as clear cut as beyond the top 100, that is nevertheless significantly less common than the top 10 tags. Considering those tags also experience a drop off in quality by PPTA, the relative LCTP perspective also suggests the top 50 is a reasonable threshold for prominent tags.

The choice of threshold for prominent tags was ultimately made by human judgement. Although data was gathered, examined and used as the basis of said judgement, nevertheless caution is warranted as human judgement is imperfect. This is a potential threat to validity of the following analysis based on prominent tags. A more rigorous and mathematically sound examination of the data may find a different threshold is more appropriate, or even that the current threshold is thoroughly insufficient for unforeseen reasons. Either such finding would weaken the value of the next section.

Furthermore, an additional threat to validity is the changing usage of tags over time. As various tags become more or less popular over time, it is possible that the ideal threshold for prominent tags will also change over time. This analysis does not account for that possibility.

2.1.2 Case Study of Question Tagging in Practice

The question in Figure 2 is an obscure, unanswered question that demonstrates some of the tagging issues in unanswered questions. The text of the question is as follows: "I am trying to implement Netflix Zuul for load balancing traffic between



Netflix Zuul-AWS Integration

I am trying to implement Netflix Zuul for load balancing traffic between different ELB's in different regions. Can somebody suggest good tutorial or any alternative method for load balancing ELB traffic.

Figure 2: An example of an unanswered Stack Overflow question that demonstrates some common tagging problems [13]

different ELB's in different regions. Can somebody suggest good tutorial or any alternative method for load balancing ELB traffic." Note particularly the second sentence, which indicates that the question asker's true intent is load balancing web traffic across regions. It also makes no mention of what technology is being used to implement this or what the intent of the task is.

While there are a number of issues with this question that are likely hindering it from being answered [14], the focus here will be on how the tags hinder discoverability. Particularly, consider that even a non-expert who stumbles upon the question could leave a comment directing the asker on how to improve this question, at which point many of the other issues will be solved or at least greatly reduced. This type of suggestion is a common practice on Stack Overflow [15]. However, that cannot happen if the question is not discovered, and tags are one of the most important components for discoverability.

The tags for this question are amazon-web-services (64 thousand questions [3]) and netflix-zuul (772 questions [3]). In this case, the tags are simply a list of the services involved in the question title. Notably, neither tag used is a prominent tag, as per the definition of a prominent tag as any tag among the top 50 most commonly used tags on Stack Overflow. There are also no tags specifying what technologies are being used with the services, nor any tags suggesting that the true question is about load balancing web traffic. There are other tags that could be added that are relevant to the question. For example, the load-balancing tag is relevant and adds to the question by showing what specifically about those technologies is in question.

For general tags, the tags dedicated to programming or markup languages are among the most popular on Stack Overflow. In the top 10 most common tags, 7 are dedicated to a specific programming or markup language. In the top 25 most common tags, 15 are dedicated to a specific programming or markup language. In the top 50 most common tags, 22 are dedicated to a specific programming or markup language.

Presumably, the asker is using some programming or markup language in order to use both of those services for their project. If the asker adds that tag to the question, it is plausible at least some of the people following that language tag have had a similar issue and could point the asker in a useful direction. Most importantly, it would bring much more attention to the question. If the asker's project is done in Java, for example, then there are 1.4 million questions with that tag [3] The level of attention to that tag is more than an order of magnitude greater than the next most popular tag used in the question (amazon-web services, with 64 thousand questions).

Adding those two tags by themselves would be a significant improvement to the question's tagging, possibly enough to bring someone to the question who can either answer it outright or direct the asker on how to improve the question further.

Questions like this which make all three tagging mistakes and remain undiscovered are rare. It is much more common for a question to make one or two tagging mistakes rather than all at once. Even among questions that do make all three, they often will still be discovered and answered eventually. However, "eventually" is not enough. The median response time for a question is 15 minutes [5], and this fast response time is one of the most important factors in the success of Stack Overflow [8]. Furthermore, the distribution of response times has a long tail that drops off significantly after the first hours. If a question does not grasp immediate attention, it is reasonable to expect multiple days to pass before it is answered. This can be seen in the *average* question response time, 2 days and 10 hours [8], which is heavily skewed by the outlier questions that are not swiftly answered. Therefore, it is desirable to improve discoverability even for questions that may still eventually be answered, as that can make a very large difference in the expected response time.

2.2 Assisting Human Tagging

Encouraging higher quality tagging on Stack Overflow questions is expected to reduce the number of unanswered questions and expedite answers in general. Tags are human generated by the question asker, so the most obvious approach is complementing their tagging process. That raises the question: What hinders human tagging?

An obvious cause is sheer volume: At the time of writing, there are 46,300 unique tags (i.e. has not been marked as a synonym for any other tag) on Stack Overflow that have been used for at least 5 questions. It is virtually impossible for a typical Stack Overflow user to know all or even most of those tags, much less efficiently select which ones are relevant to their question.

Broadly speaking, there are two ways to address a data volume problem such as this. The first is "searching," by providing some algorithm that can efficiently identify which results are relevant for a given query. In this case, the question itself is the query, and the algorithm seeks to find which tags are relevant.

The other approach is "sorting," by providing some efficient organization of the dataset such that manually finding a desired element or set of desired elements is fast and simple for the average human with their existing knowledge. This approach is somewhat more challenging to apply in this case due to the question of what criteria should be used to organize the data? For example, an obvious way to sort the tags would be in alphabetical order. Although that does make it simple to find a tag on a list assuming the name of the tag is known, it is useless for this problem because the user does not know the name of the tag they desire. What the user does know is the domain knowledge of their problem and the general scope and technologies involved.

A potentially useful organization, then, is a hierarchy of tags, with more specific tags as the leaves of more general tags. It is likely there is a tag named directly after the technology involved in the question and said tag can be readily identified by the user. This is evidenced by tags named after technologies being among the most widely used tags on Stack Overflow: 46 of the 50 most commonly used tags on Stack Overflow are named after a technology of some kind. From there, the hierarchy might point the user towards more specific tags for that technology, among which may be tags relevant to their specific use case. Alternatively, if the user starts with a very specific technology, the hierarchy could point the user towards more general tags encompassing the user's technology but are still relevant to the question. Existing evidence suggests a hierarchical tag organization is beneficial for both contributing and for reading [4] [16] [17]

Both of these are viable approaches to improving tagging quality. The remaining question is what algorithm could efficiently identify relevant tags and present them to the user, and what algorithm could organize the existing tags in a hierarchy of general and specific tags.

CHAPTER 3

RELATED WORK

When someone asks a question on a site like Stack Overflow, the question itself typically has a reasonable sized body of text detailing the subject of the question. The user is likely to put a decent amount of effort into writing the question in effort to explain the question effectively. Given these characteristics, the body of text appears to be a prime input for some natural language processing (NLP) algorithm which can then predict whether a tag is likely to apply to that particular question body or not.

There are a number of challenges with that, however, especially the fact that questions on Stack Overflow aren't truly natural language text. Many questions on Stack Overflow will include some amount of code mixed in the question body, which may or may not be separated from the normal language text. The code follows a very different set of rules and grammar than natural language, which could be a challenge for existing NLP algorithms. Certain words may have one particular meaning in natural language but mean something entirely different in the context of code.

An example of this problem is the word "for". In English, the word "for" is a preposition that indicates purpose. In many NLP problems, "for" provides so little information that it is filtered out in preprocessing, and doing so will often improve the accuracy of the NLP model. [18] By contrast, in the C programming language, "for" is a keyword that indicates a specific type of iterative loop, and provides vital information about what is happening in the program.

This issue is further complicated by the fact that there are many possible programming languages that could be used in a Stack Overflow question, each with their own syntax and keywords, which may or may not be shared across languages and may or may not mean the same thing in different languages. The keyword "for" may refer to a specific iterative loop in the C programming language, but in the Java programming language, it might refer to that loop style or is may refer to a different so-called "for-each" loop that examines each element in a specific data structure, depending on how the keyword was used. Other languages often also include the "for" keyword, which often means still different things and has different usages in those languages.

This is a significant challenge for a number of traditional NLP techniques and will require special care to handle properly. Unfortunately, since code is such a vital component to many Stack Overflow questions, it is not safe to simply ignore it. Nevertheless, despite the challenges, it appears there should be some method in this style that could predict tags from the question body.

The other question is how to create the tag hierarchy. There is no information in the tags themselves that makes it obvious how they are related. By examining how the tags are used with various questions, it should be possible to infer how tags are related. A very simple example of this is noticing that the SWING tag is rarely used without the JAVA tag, but the JAVA tag is often used without the SWING tag. From that, it is reasonable to infer that SWING is a more specific element of JAVA.

By itself, this type of intuition is not enough. This simple example could be formalized by setting some thresholds and comparing tag occurrences and co-occurrences to the thresholds. Unfortunately, doing so would produce a directed graph, not a hierarchy. Mathematically, a directed graph is still a valid and useful organization for tags, but such an organization no longer has the intended clear path between specific and general, and is arguably less intuitive for human understanding. Therefore, it is not an acceptable solution for this issue.

It should still be possible to use a similar style of model to ultimately produce a hierarchy. It could be augmented to include some method of reducing the directed graph into a hierarchy, for example.

An alternative approach is to use NLP models with the question body to associate certain text with certain tags. The models for each tag could then be used to deduce the correct ordering for a hierarchy. While more elaborate, it's notable that this approach has potential to better capture the tag meaning and thus produce a higher quality hierarchy. It's also notable that this approach is similar to the concept for predicting new tags. If a similar model is used for both predicting new tags and generating a hierarchy, then that would create logical consistency between tag predictions and tag organization as both were generated from similar models.

From this, both questions ultimately reduce to what NLP techniques would be applicable for producing tag predictions and generating a hierarchy.

3.1 Tag Recommendation

There have been a number of models that could predict labels for some form of text document. For instance, Saha et. al. used a support vector machine for every label that each classify whether that label applies to a given document or not [19]. Fang et. al. used tensor factorization with a Gaussian kernel in a similar fashion [20]. Labeled Latent Dirichlet Allocation has been used for tag recommendation [21] [22]. All of these methods are based on non-trivial machine learning models and/or mathematics. While each method has merits, they have the drawback of obfuscating some of the particular nuances of the problem.

3.1.1 Hierarchical Clustering Models

A hierarchical clustering model is one of the simplest models that can potentially be effective at tag recommendation [23] [24]. This model divides a set of documents into clusters using some unsupervised machine learning technique. A range of techniques could be used for forming the clusters, including both divisive and agglomerative (top-down division and bottom-up merging) techniques. Regardless of clustering technique, all resulting clusters are organized in a hierarchical fashion, with clusters at the root being more general and encompassing more documents, while those at the leafs are small, specific clusters, only including a small number of documents.

Once all documents have been assigned to a cluster, then documents within the same cluster could be considered to share some intrinsic similarity. In the context of the Stack Overflow problem, this type of model has some appealing properties. The clusters produce groups of documents that share some intrinsic similarity, which is very similar to what the concept of a tag is supposed to be. These clusters are also organized in a hierarchy that goes from very general to very specific, addressing one of the key issues mentioned in the introduction.

Despite this, the model falls short for this problem for several reasons. While the clusters achieve a similar purpose to tags, there is not necessarily any correspondence between an existing tag and a learned cluster. It's even possible that the cluster has no simple human understandable meaning, and is thus entirely irrelevant to this problem. Furthermore, because the model is unsupervised, it has no way of taking the existing tags into account while training. Even if by coincidence the clusters do end up corresponding to tags, there will be no way to determine which clusters map to which tags beyond manually examining and labeling them. There has been some work to solve this, however. Some variants allow some specificity in how clusters are formed, meaning it would be possible to use tags as the groups for clustering [25] [26].

However, there is another significant issue with using hierarchical clustering for the Stack Overflow tagging problem. Each document is only assigned to a single cluster path, which creates an issue for documents with multiple tags. Even if each cluster directly corresponded to an existing tag in Stack Overflow, this model would only be able to handle documents if all tags were in the same cluster path. For example, this question [27] is about interaction between python and java code, and is tagged with both the PYTHON and JAVA tags. Given that most JAVA and PYTHON questions are about issues within their respective languages, it is unlikely that a sane hierarchy will place them in the same hierarchy path: JAVA is not a generalized issue that encompasses PYTHON, or vice-versa. Meaning, since these tags belong to disjoint positions in the hierarchy, the hierarchical clustering model is fundamentally incapable of handling this question correctly.

3.1.2 Probabilistic Topic Models

Another type of model that could potentially be useful for tag recommendation is a probabilistic topic model. In a probabilistic topic model, there are a number of groups, called topics, and each document is assigned some probability of belonging to each topic. These models are applicable to any type of discrete data and is especially popular for NLP tasks. This type of model is interesting for this problem because it is fundamentally capable of assigning any document to any number or combination of topics.

Like hierarchical clustering, probabilistic topic models are typically unsupervised. For example, Latent Dirichlet Allocation (LDA) is reasonably well used topic model and it is unsupervised, and a labeled variant of it has previously been used for this problem [21] [22]. In a typical unsupervised topic model, the topics are learned rather than specified, similar to how clusters are learned rather than specified in hierarchical clustering. For this problem, that brings all the same issues as hierarchical clustering had.

Additionally, some topics models offer methods to ensure tags would be meaningfully associated with the inferred topics, even if the model is fundamentally unsupervised. Between this, probabilistic topic models avoid both of the significant issues with hierarchical clustering.

While it solves those problems, there are also some new issues with probabilistic topic models that didn't exist with hierarchical clustering, the most notable of which is the absence of a hierarchy. Without the hierarchy, there is no clear organization of topics in terms of specificity or generality. Even outside the context of this particular problem, a hierarchy could be desirable because it makes it simpler to interpret topic meanings, particularly when the topics must be learned in an unsupervised model.

3.1.3 Hierarchical Topic Models

Hierarchical topic models work similarly to other probabilistic topic models, but include some algorithm for organizing the learned topics in a hierarchy. On the surface, this appears to combine the strengths of a probabilistic topic model with the strengths of hierarchical clustering. However, many hierarchical topic models still maintain some of the attributes that make hierarchical clustering or probabilistic topic models unsuitable for the Stack Overflow problem.

A number of hierarchical topic models have been developed, such as the hSLDA model [28] and the hLLDA model [29]. Similar to regular probabilistic topic models and hierarchical clustering, hierarchical topic models are typically unsupervised. A number of models, such as hierarchical LDA (hLDA), are only able to assign documents to one topic in the hierarchy. Multiple other models have been proposed to address this and more issues, but still many maintain attributes unsuited for this application.

One hierarchical topic model in particular, L2H [6], seems to address many of the concerns with hierarchical topic models that are relevant to the Stack Overflow problem. In particular, L2H is a supervised model, requiring a training dataset from which it explicitly creates one topic for every tag. By doing so, L2H avoids any of the issues with dissociation of learned concepts from existing tags.

While training the topics, L2H also builds a graph of the "concepts," which is L2H's term for tag/topic pairs. At the end of training, a hierarchy is built from the graph organizing the concepts from most to least general. Notably, however, the graphs weights are updated during training. This means associations between concepts are not limited to estimating tag associations based on how the initial training data is tagged.

While that is used as an initial graph, L2H also learns new associations between concepts during training even if the initial tags do not make it obvious [6]. This is an important feature because if the original tagging is neither entirely complete nor entirely correct, as in many human generated data sources, then L2H can still learn from it. A well-trained L2H model can even predict novel tags for training data that may have been absent, or identify tags in the training data that may be inaccurate.

L2H appears to combine the strengths of probabilistic topic models and hierarchical clustering while including very few of the attributes that are undesirable for this problem. It provides an immediate answer to the issue of organizing tags in terms of specificity with the hierarchy, while also providing an answer for tag recommendations with the topic model. Unlike hierarchical clustering, a document isn't assigned to one specific path in the hierarchy, but rather is assigned probabilistically, avoiding the issue of a document fitting in multiple paths. Also unlike hierarchical clustering, the tags directly correspond to a component in the model, providing little chance of a mismatch between tag and model. Finally, an additional bonus is the fact that L2H does not assume the training data is entirely correct, making it especially suited for the Stack Overflow data.

CHAPTER 4

METHODOLOGY

In this chapter, the details of how L2H works will be examined. The difficulties involved in applying L2H to Stack Overflow will then be discussed, followed by a discussion of how additional tag synonym predictions can be generated from the model.

4.1 L2H

For a given tag, every word has a certain probability of suggesting that tag is relevant. This set of probabilities forms a discrete probability distribution, called a topic, that is associated with that tag. The pair of tag and topic is referred to as a concept. Concepts are organized internally as a weighted directed graph, with each concept as an edge in the graph. If a document in the training set is tagged with two different tags, then an edge is added between them with the weight equal to the number of documents tagged with both tags versus the number of documents with the starting tag alone. [6]

More formally, words are the most basic discrete unit of data under consideration by L2H. Let w denote a word. A document is defined as a sequence of words $d = \{w0, w1, w2...wN_w\}$ where N_w is the number of words in the document and $w_i, 0 \leq i \leq N_w$ is a word in the document. A corpus is a collection of documents $D = \{d0, d1, d2...dN_d\}$ where N_d is the number of documents in the corpus and $d_i, 0 \leq i \leq N_d$ is a document in the corpus. For a given corpus, the set of all words that occur at least once is the vocabulary, denoted as $V = \{w0, w1, w2, ...wN_v\}$, where N_v is the number of words in the vocabulary and $w_i, 0 \leq i \leq N_v$ is a word in the vocabulary.

A tag is a word that can be associated with specific documents. Let t denote a tag. For any given document, there is a non-empty set of tags associated with it, denoted L_d.

For a given corpus, the set of all tags that occur at least once is the label vocabulary, denoted as $L = \{t0, t1, t2...tN_l\}$ where N_L is the number of tags in the label vocabulary and $t_i, 0 \leq i \leq N_l$ is a tag in the vocabulary.

Over a given corpus's vocabulary, V, and label vocabulary, L, topics and concepts can be defined. A topic is a discrete probability distribution with the probability mass function $P_\phi(w = w_i) = P_{\phi,i}$ where $0 \leq i \leq N_v$ and $\sum_{i=0}^{N_v} P_{\phi,i} = 1$. Every topic is associated with a tag, and there are no topics that are not associated with a tag. That is, if Φ denotes the set of all topics, then the number of topics in set Φ equals the number of tags in the label vocabulary, N_L. There also is a one-to-one mapping between topics and tags. For this, the notion of a concept is introduced. A concept is defined as a pair (l, ϕ) where $(l \in L)$ is a tag and $(\phi \in \Phi)$ is a topic.

The task of L2H is to take a corpus of documents and the associated vocabulary and label vocabularies and construct a hierarchy of concepts. Let $G = (V, E)$ be a weighted directed graph. Each vertex in this graph corresponds to a concept. The initial weight for each edge from vertex V_i to V_j is the number of documents tagged with both tag t_i and tag t_j, denoted $D_{i,j}$, divided by the number of documents tagged with tag t_j, denoted D_j so $weight(V_i, V_j) = D_{i,j}/D_j$. Finally, a background node is added to the documents with zero initial weight on the edges going towards the background node, so $weight(x, background) = 0$, where x is any vertex in V. The weight from the background node to every other node equals the number of documents tagged with tag t_i, denoted D_i, divided by the number of occurrences of the most common tag, so $weight(background, V_i) = D_i/max_k(D_k)$, where $max_k(D_k)$ is the number of occurrences of the most common tag.

After the initial graph is generated, a Markov Chain Monte Carlo (MCMC) algorithm is used to infer the final hierarchy from the training data [30]. The number of possible hierarchies increases rapidly with the number of tags present. To practically perform inference on large datasets, it is essential to select a good prior hierarchy.

The prior hierarchy is constructed by using Chu-Liu/Edmonds' algorithm [31] for the maximum spanning tree on graph G. The background node is used as the root.

4.1.1 Switching Probability

The basic principle for learning in the L2H model is similar to that of the LDA model. Each word in a document is considered to be generated by one of its topics. The set of tags associated with a document, L_d indicates which topics are more likely to be used. Defining the words of a document like this creates focused topics [32]. However, it is not safe to assume the set of tags associated with a document is complete. It is likely that users overlooked some tags. Furthermore, in the case of Stack Overflow, users are limited to 5 tags per question. A question which has more than 5 applicable tags is technically impossible to tag correctly.

To address this issue, there are two sets of tags used during learning. For each document, the tags are divided into two subsets, L_0 and L_1. L_1 includes both the document's inherent tag set, L_d, and all tags along the path from the root to any tag in L_d. L_0 is the complement of L_1, containing all tags in L but not in L_1.

L_1 is defined as such because it captures relevant broader tags. For example, if a question is tagged with ANDROID-TOOLBAR then logically the tag ANDROID should also be applicable because ANDROID is a superset of ANDROID-TOOLBAR. Defining L_1 with this broader information ensures it captures tags that should be relevant given the human generated tag set, L_d. In contrast, this ensures L_0 captures all tags that there is no a priori reason to believe are directly relevant to the document. However, there may be relevant tags to be discovered in L_0, and it is possible some tags in L_1 are actually irrelevant. This is the same issue of imperfect tagging mentioned earlier, but this more formal definition is amenable to a solution.

Let $\gamma = (\gamma_0, \gamma_1)$ be a hyperparameter. For each document, a switching probability, π_d, is defined by a draw from the distribution $\pi_d = Beta(\gamma_0, \gamma_1)$. The switching probability specifies how likely a given token is to be generated by tags in L_1 as opposed to tags in L_0. This avoids a strict requirement that all words in the document are generated by one of the human labeled tags or their related tags. Instead it only maintains a soft preference for those tags and leaves open the possibility of incorrect human tags or undiscovered additional tags. This is how L2H avoids assuming that documents are tagged exhaustively.

Thus, the general procedure for sampling topics in L2H looks like this for each document:

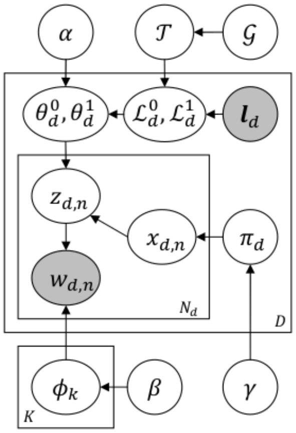
1. Assign a switching probability $\pi_d = Beta(\gamma_0, \gamma_1)$ to the document
2. For each token, randomly decide whether the label set will be L_1 or L_0 with probability π_d, then randomly select a tag/topic pair from that label set.
3. Compute the conditional probability of a word given the token via bottom-up smoothing followed by top-down sampling [33] and update the topic accordingly.

Once the topics are updated, then the tree structure can be updated accordingly. This is done via the Metropolis-Hastings algorithm [34]. For each vertex V_i in the graph except the background node, randomly select an incoming node V_k that is not currently considered a descendent node, with the probability weighted proportionally by edge weights. Randomly decide whether or not the new node V_k is assigned as the parent of node V_i.

Overall, the functionality of L2H can be summarized by Figure 3, as presented in the original paper introducing L2H [6].

4.2 L2H on Stack Overflow

Applying L2H to Stack Overflow presents a number of challenges. The mixture of natural language text with source code is a unique format that does not necessarily lend itself to existing NLP techniques [35] [36] [37]. Additionally, human error is a significant



1. Create label graph $\mathcal{G}$ and draw a uniform spanning tree $\mathcal{T}$ from $\mathcal{G}$ (§ 2.1)
2. For each node $k \in [1, K]$ in $\mathcal{T}$
   (a) If k is the root, draw background topic $\phi_k \sim \text{Dir}(\beta \boldsymbol{u})$
   (b) Otherwise, draw topic $\phi_k \sim \text{Dir}(\beta \phi_{\sigma(k)})$ where $\sigma(k)$ is node k's parent.
3. For each document $d \in [1, D]$ having labels $\boldsymbol{l}_d$
   (a) Define $\mathcal{L}_d^0$ and $\mathcal{L}_d^1$ using $\mathcal{T}$ and $\boldsymbol{l}_d$ (cf. § 2.2)
   (b) Draw $\theta_d^0 \sim \text{Dir}(\mathcal{L}_d^0 \times \alpha)$ and $\theta_d^1 \sim \text{Dir}(\mathcal{L}_d^1 \times \alpha)$
   (c) Draw a stochastic switching variable $\pi_d \sim \text{Beta}(\gamma_0, \gamma_1)$
   (d) For each token $n \in [1, N_d]$
      i. Draw set indicator $x_{d,n} \sim \text{Bern}(\pi_d)$
      ii. Draw topic indicator $z_{d,n} \sim \text{Mult}(\theta_d^{x_{d,n}})$
      iii. Draw word $w_{d,n} \sim \text{Mult}(\phi_{z_{d,n}})$

Figure 3: Graphical representation and description of L2H's behavior, as presented in [6]

factor among Stack Overflow text. While L2H has inherent mechanisms for addressing the human error in tagging, the document text itself is likely to contain various forms of human error. For instance, not all words will be spelled correctly. Sentences will not necessarily use valid grammar. Vital details may not be explicitly written or stated in the question. Answers may be outright incorrect or irrelevant.

Notably, some context and information may be left out if it is reasonable to expect a human reader to understand that information a priori, but the algorithm does not necessarily have that information. While this is true for many NLP problems, it is especially prevalent in technical Q&A text such as Stack Overflow. For instance, the term "query" will likely have a very different meaning when used in a question about SQL (probably referring to a database lookup query or how to express something in the SQL syntax) than it will in a question about jQuery (probably referring to either the technology itself or how to use the API to access particular elements of a webpage). If the text does not make it explicit which variant of the term "query" is being referred to, it is difficult for an algorithm to infer the meaning of the term. That is particularly problematic when determining whether or not to tag a question with the SQL tag or the JQUERY tag.

To some extent, L2H accounts for this. Probabilistic topic models such as L2H can model uncertainty in their probabilities. Nevertheless, effectively preprocessing the documents can have notable impact on the overall performance of the model and is therefore an important consideration. There are three significant preprocessing decisions for consideration.

4.2.1 Preprocessing Considerations

1. What text should constitute a document?

2. Which questions should be included in the training corpus for L2H?

3. What steps are necessary to adequately model the mixture of both code and natural language text?

The first significant preprocessing decision is what text should constitute a document? Possible choices include the question text by itself, the text of the answers if present, the text of the comments, whatever text is exclusively natural language text or exclusively code text, etc. Consider that one of the key purposes of applying the model is to potentially provide tag suggestions while a user is asking a question. In such a scenario, the model will only have access to the question text. Additionally, the tags for a question are supposed to be decided based on their relevance to the question itself, not necessarily how the question was answered, so it makes theoretical sense to use just the question text. Given that, the question text is used by itself as the text for L2H.

The next significant preprocessing decision is which questions should be included in the training corpus for L2H? While it is theoretically possible to use the entirety of Stack Overflow as the training corpus, the computational cost would be prohibitively expensive. Furthermore, not all questions on Stack Overflow are necessarily quality training samples. The text of a question that was closed for being "not a question," for example, is unlikely to actually state any form of question, and therefore is unlikely to be a useful training sample for the model. Likewise, questions that were moved to other Stack Exchange sites may not be representative samples for questions that belong on Stack Overflow.

In particular, however, questions with primarily uncommon tags are unlikely to contribute to the model. 26.78% of tags occur less than 10 times. Such tags are almost certainly difficult or impossible to build a valid and usable model for, and are of dubious usefulness to the dataset as a whole. Furthermore, only 7.31% of tags occur more than 1,000 times, suggesting that a relative minority of the tags on Stack Overflow constitute the majority of tag occurrences.

To take advantage of these insights, questions are selected in small batches starting from a commonly occurring tag such as JAVA or ANDROID. This ensures all questions have at least one non-rare tag, and thus provide value to training the model. However, it is still possible to include a question with a rare tag if such a tag occurs alongside a common one. To avoid rare difficult tags, all tags that occur below some minimum label frequency are filtered out and not used by the model. The end result is a usable training corpus for building the model.

The final significant preprocessing decision is how to adapt for the mixture of both code and natural language text. One seemingly obvious tool is the <code> html tag which is supposed to be used in questions to format code blocks. Unfortunately, this html tag is not consistently used to mark code, instead being used primarily when special code formatting is desired, even if the text itself is natural language instead of code. Therefore, it is not a reliable indicator of what is code and what is natural language text. Nevertheless, it is a useful approximation and is used here to separate code from natural language.

To properly represent both code and natural language, they are initially separated by the code tag. Natural language text is divided into words using spaces and punctuations as word boundaries, and frequent bigrams are merged and treated as a single word. Any excessively rare or excessively common words are filtered out as not useful for the model. Code is divided into words with a similar process, accounting for the fact that valid symbol boundaries in natural language text are not necessarily valid boundaries in code text. A similar process of merging common bigrams and removing overly common or rare occurrences is also performed. Once this is finished, the sets of words from the code and natural language portion together form the words for the document.

With this, the initial corpus is selected, fully preprocessed, and ready for use in training the model. While this preprocessing does not account for all known difficulties with the Stack Overflow dataset, it does account for a practical subset of them. Additionally, as discussed earlier, L2H has inherently useful properties for some of the challenges in the Stack Overflow dataset. The current preprocessing should enhance those properties and thus indirectly reduce the significance of those issues.

4.2.2 Hyperparameter Selection

In building L2H on this training dataset, hyperparameters must be appropriately chosen. L2H has α, β, and γ hyperparameters as illustrated in Figure 3. All of these hyperparameters are a form of prior: as the amount of training data increases, the influence of the hyperparameter values on the final model decreases. The Stack Overflow dataset, having tens of millions of unique posts is very large. Although the training data is notably smaller than this, it is still a rather large number of training samples. Given this, the recommended default hyperparameters from the segan library are used. The expectation is both that segan's default hyperparameter choice is reasonable for this model and that hyperparameter choice will ultimately have minimal impact on the final model. This expectation is supported by existing work [38]. It was further tested in practice by building models on a separate sample and confirming that the resulting models were well fitted via validation.

4.3 Tag Synonyms

Once the model is run on the training dataset for a number of iterations, it can be validated in a number of ways. A topic model is most commonly validated by using perplexity or predictive likelihood [39]. Model validation can be used to select reasonable thresholds for filtering during preprocessing and ensuring hyperparameters are reasonable.

An additional validation tool can be used in this case by using tag synonyms. Stack Overflow has an existing dataset of human labeled tag synonyms created by existing users of Stack Overflow. The finished model can be used to predict whether or not two particular tags are synonymous and compare that prediction against the human labels. If L2H's tag synonyms are significantly different from human identified tag synonyms, that indicates that L2H's model of the tags is notably different from human understanding of the tags. In such a case, even if L2H otherwise performs well on predictions, the gap between the model's understanding of tags and human understanding of tags suggests it would not be useful for recommending tags to humans, failing one of the primary purposes of applying L2H on Stack Overflow.

4.3.1 Topic Distance

L2H does not have an inherent mechanism for testing if topics are similar, so an appropriate method must be determined. Once the L2H model is built, it has a tag/topic pair, (l, ϕ), for every tag in the corpus. It is common to compare two topics using the Kullbach-Leibler divergence (KL-Divergence) [40] [41] [42]. KL-Divergence is defined as $D(x, y) = -\sum_{i=0}^{n} x(i) log(\frac{y(i)}{x(i)})$, where x and y are discrete probability distributions of size n.

KL-Divergence is a desirable measure due to its roots in information theory and probability, and it lends itself to theoretically sound methods for comparing two topics. However, it is not a suitable measure for identifying tag synonyms. A basic property of synonyms is they are symmetric: If A is a synonym of B, then B is also a synonym of A. KL-divergence, by contrast, is not a symmetric measure. KL-Divergence of A to B can give a very different number than KL-Divergence of B to A. What is instead desired is a metric with similar information theoretic and probabilistic properties that allow theoretically sound comparison, but is also symmetric.

An appropriate metric, then, is the Jensen-Shannon divergence [43] [44]. Intuitively, in desiring a measure with properties of KL-Divergence but with the addition of symmetry, one might try simply taking the KL-Divergence for both possible directions. For example, $f(x, y) = \frac{1}{2}D(x, y) + \frac{1}{2}D(y, x)$. This does not suffice because it is possible for $D(x, y)$ to be infinity if $y(i)$ is zero but $x(i)$ is non-zero.

That problem can be avoided by using a mixture distribution. Let $M(x, y) = (x + y)/2$ be a mixture distribution for probability distribution x and probability distribution y. A notable property of $M(x, y)$ is if either $x(i)$ or $y(i)$ is non-zero, then $M(x, y)(i)$ is also always non-zero. Given this, a function like $g(x, y) = \frac{1}{2}D(x, M(x, y)) + \frac{1}{2}D(y, M(x, y))$ will be symmetric and based on KL-divergence, but will no longer have issues of infinite values. In fact, it is possible to show $0 \leq g(x, y) \leq log(2)$ for all x, y. $g(x, y)$ is, in fact, the definition of the Jensen-Shannon divergence. That is, $JS(x, y) = g(x, y)$. Viewed like this, one can intuitively see how the Jensen-Shannon divergence preserves the desirable properties of KL-Divergence for this problem while resolving the undesirable complications of asymmetric behavior and infinite values.

A tag synonym can then be identified by taking the Jensen-Shannon divergence of two topics. Below some threshold, the topics are considered to be synonyms, while above that threshold they are considered to not be synonyms. This is how L2H's predictions of tag synonyms are determined and ultimately compared with human identified tag synonyms for validation.

4.3.2 Hierarchy Integration

It is notable that topic distance approaches to identifying tag synonyms makes no usage of the hierarchy inferred by L2H, only the tag topic distributions. As L2H also generates the hierarchy, and the hierarchy is supposed to provide useful information about the tags and their relationship, it is logical that integrating the hierarchy into the predictions should improve the ability to identify tag synonyms. However, the correct way to integrate the hierarchy into the existing synonym identification scheme is not necessarily obvious.

Intuitively, tags that are closer to each other in the hierarchy are more likely to have related meanings. From this, a basic technique to utilize the hierarchy is with graph distance. Let $\delta(\phi_0, \phi_1)$ be the shortest graph distance between two tag topics in the hierarchy. Note that this graph distance is across the final generated hierarchy, not across the weighted directed graph used to generate the hierarchy.

With the δ function, a simple threshold comparison can be done: If δ is above a certain threshold, then the topics are considered to be too far apart to be synonymous. This is logical because topics that are very far apart in the hierarchy should be conceptually unrelated, and therefore not synonyms even if the topic distribution may look similar. Thus, overall, a tag can be considered synonymous if and only if $\delta(\phi_0, \phi_1) \leq \epsilon_0 \wedge JS(\phi_0, \phi_1) \leq \epsilon_1$, where ϵ_0 is some threshold for maximum graph distance and ϵ_1 is some threshold for maximum Jensen-Shannon distance.

This method of identifying synonyms is suboptimal. More generally, $\delta(\phi_0, \phi_1)$ and $JS(\phi_0, \phi_1)$ can be viewed as features for some unknown classifier, C, which is capable of classifying an input as synonymous or not synonymous based on the input features. Viewed as this, the issue of identifying tag synonyms becomes a separate machine learning problem where the objective is to identify the best possible C for identifying tag synonyms. While in theory any reasonably sound classifier could be used for C, it is notable that the input features are ultimately based on L2H.

That reasoning can be taken a step further. The issue is then, why are $\delta(\phi_0, \phi_1)$ and $JS(\phi_0, \phi_1)$ specifically the features? Viewed even more generally, C is testing for internal redundancy in the model generated by L2H: if two topics are truly synonymous, then learning separate representations for them is redundant. In fact, having such redundancy reduces the power of the model because each of the synonymous representations lost some potential training material to the other redundant topic.

Therefore, the internal state of L2H should be the feature set for C. Furthermore, the classification scheme used by C should be based on the internal workings of L2H to most accurately determine whether L2H is learning a redundant topic. In other words, C is actually an extended component of L2H itself for checking redundancy.

Extending L2H with a redundancy checking component is a potentially useful direction for future research. However, it is also outside the scope of this work.

For the purposes of this work, then, not all potential methods of identifying tag synonyms will be able to be tested. Overall, four methods of identifying tag synonyms have been proposed thus far.

1. $JS(\phi_0, \phi_1) \leq \epsilon_1$
2. $\delta(\phi_0, \phi_1) \leq \epsilon_0 \wedge JS(\phi_0, \phi_1) \leq \epsilon_1$
3. Classifier C with features $\delta(\phi_0, \phi_1)$ and $JS(\phi_0, \phi_1)$
4. Extend L2H with internal redundancy component

The cutoff decision can be viewed in terms of the complexity added by testing the next method.

The additional complexity in method 2 compared to method 1 is a new measure, $\delta(\phi_0, \phi_1)$ must be computed and a new threshold, ϵ_0, must be determined. $\delta(\phi_0, \phi_1)$ is computed by calculating shortest graph distance on the existing hierarchy.

The additional complexity in method 3 compared to method 2 is a new classifier, C, must be trained with the existing features. Some machine learning model must be selected for C. That may involve training and testing several models for C before determining the best one, for some criteria. Furthermore, if C is based on a supervised machine leaning model, then some training data must be created or generated in order to build C.

The additional complexity in method 4 compared to method 3 is L2H must be further studied for additional understanding of its behavior. Then an appropriate model of that behavior for the purpose of identifying redundant topics must be identified and captured, and said model should have theoretical grounding in L2H's internal behavior and accurately determine when redundant topics exist in general.

In terms of added complexity, methods 1 and 2 can be obviously tested without going extensively out of scope for this work. Method 3 could theoretically be tested, particularly if there was a clear candidate model for the task. In absence of such a model however, the process of testing and identifying models to determine the best tool for distinguishing tag synonyms is an involved enough process to warrant its own research, and thus is not performed in this work. Therefore, methods 1 and 2 will be tested and their performance for identifying tag synonyms will be compared. Methods 3 and 4 will not be tested as out of scope for this work.

CHAPTER 5

EVALUATION

In this chapter, several criteria for evaluating the effectiveness of L2H on Stack Overflow will be defined. The results of several evaluation tests will be examined and used to answer the evaluation criteria, providing an overall answer to L2H's effectiveness on Stack Overflow.

Evaluating the model centers on questions related to the two key criteria defined and discussed throughout the paper. Namely, the issue of creating a hierarchy organizing tags from general to specific and creating a model that can recommend useful tags for a human tagger. Additionally, the effectiveness of tag synonym prediction can be examined as a potential additional benefit for the model. Overall then, the evaluation criteria are:

1. Does L2H produce a hierarchy organizing tags from general to specific?
2. How effective is the hierarchy in finding related tags from known tags?
3. Does L2H predict similar tags to human chosen tags on unseen questions?
4. How effective is L2H at recommending tags for human tagging?
5. How effective is L2H at predicting tag synonyms?

All of these criteria require a L2H model. To create a usable model for evaluating these criteria, all Stack Overflow posts between January 1, 2016 and March 13, 2017 are selected as prospective input data. Any tags used in more than 5,000 questions or fewer than 1,000 are removed. This removes rare tags or cases of excessively abundant tagging that would not be statistically distinct enough to build a usable model for. Note that the threshold is balanced to avoid filtering out the most popular Stack Overflow tags solely by virtue of being popular. If there is enough data to plausibly distinguish them from other topics, then they will be included in the model as well.

Words are filtered out of questions if they appear in 40% or more of questions, or if they appear fewer than 300 times. This is the same filtering step discussed in the methodology section. A separate model was produced on a separate sample to verify the quality of these thresholds.

After the remaining preprocessing is performed as detailed in the methodology section, the resulting dataset contains 369 tags and 196 pairs of tag synonyms according to Stack Overflow's tag synonyms page [45].

The model hyperparameters are set at $\alpha = 10$, $\beta = 1000$, and $\gamma = (\gamma_0 = 0.9, \gamma_1 = 0.1)$. The model was then trained on the dataset and used for the remainder of the evaluation.

5.1 Exploratory Search Effectiveness

One of the stated goals for applying the L2H model to Stack Overflow is providing a hierarchy that is useful for exploratory searches. To do so, the hierarchy must score well on evaluation criterion 1, 2, and 3.

Evaluation criteria 1 asks whether the hierarchy does in fact organize tags from general to specific. A potential method of answering this is by using the concept of prominent tags introduced in the background section. Unfortunately, prominent tags only provide a binary classification of whether a tag is prominent or not prominent. Such an evaluation would only be effective at levels of the hierarchy where the tags tended to cross the threshold. It provides little insight into whether the tags continue to become more general or more specific at other locations in the hierarchy. Furthermore, prominent tags are intended as an approximation to the concept of general and specific and are therefore not an entirely accurate method for formally evaluating the hierarchy.

5.1.1 Specificity and Diversity

Two metrics are used to quantify how well the hierarchy organizes tags from general to specific: Specificity and Diversity [46]. Specificity is used to measure if a document is specific to its subtree. The specificity of a set of documents is expected to increase as tags become deeper in the hierarchy. Diversity is used to measure how different two sets of documents are. Diversity is expected to increase when comparing entirely different branches of the hierarchy.

Let $\theta_{m,n}$ be the weight of a concept for a particular document, where $1 \leq m \leq D$ and $1 \leq n \leq K$ and D is the number of documents and K is the number of concepts. The Shannon entropy can be estimated for each document as $H_m = -\sum_{n=1}^{K} \theta_{m,n} \log_2 \theta_{m,n}, 1 \leq m \leq D$. To illustrate that this is a useful measure for the problem at hand, consider the case where a document has equal weights for all concepts. In such a case, $H_m = -\sum_{n=1}^{K} \frac{1}{K} \log_2 \frac{1}{K} = -\log_2 \frac{1}{K} = \log_2 K$. Note that this is both the maximum value the function can have for a document and it is also the point at which the document is most diverse with the concepts. Conversely, consider a document that is entirely weighted to a single concept. In this case the limit of H_m will be considered for correct evaluation: $H_m = -\lim_{\theta_{m,1} \to 1} \theta_{m,1} \log_2 \theta_{m,1} - \sum_{n=2}^{K} \lim_{\theta_{m,n} \to 0} \theta_{m,n} \log_2 \theta_{m,n}, 1 \leq m \leq D = 0$ Note that this is both the minimum value the function can have for a document and it is also the point at which the document is the least diverse with the concepts. Thus, the Shannon entropy estimate usefully captures information about document diversity.

Let $\overline{\theta}_n$ be the average weight for a concept n across all documents, formally defined as $\overline{\theta}_n = \frac{1}{D} \sum_{j=1}^{D} \theta_{j,n}$. Let $\overline{H_m} = -\sum_{n=1}^{K} \theta_{m,n} \log_2 \overline{\theta}_n, 1 \leq m \leq D$. $\overline{H_m}$ estimates Shannon entropy for the concept averages, but still using the document's concept weights to weight the terms. It equates to the amount of entropy expected if the document weights are neutral. The difference between the true document entropy and the expected neutral entropy for the document can be defined as $V_m = \overline{H_m} - H_m$ and is referred to as the document divergence. The average document divergence across all documents is

$\overline{V} = \frac{1}{D}\sum_{m=1}^{D} V_m$ and is the primary measure of Diversity.

A similar logic applies to the definition of concept specificity, defined as $S_n = \frac{1}{D}\sum_{m=1}^{D} \frac{\theta_{m,n}}{\overline{\theta}_n} \log_2 \frac{\theta_{m,n}}{\overline{\theta}_n}, 1 \leq n \leq K$. This definition is based on the Shannon entropy estimate with each weight normalized by the average concept weight. To illustrate that this metric achieves its intended purpose, consider the case where a concept is equally present in every document such that $\theta_{m,n} = \overline{\theta}_n$. In this case, $S_n = \frac{1}{D}\sum_{m=1}^{D} \frac{\overline{\theta}_n}{\overline{\theta}_n} \log_2 \frac{\overline{\theta}_n}{\overline{\theta}_n} = \frac{1}{D}\sum_{m=1}^{D} 0 = 0$. Note that this is the both the minimum value of the function and the point where a concept is the least specific it could possibly be. Consider another case where a concept is only present in a single document. In that case, $\theta_{1,n} = 1, \theta_{m,n} = 0 for 2 \leq n \leq D$ meaning $\overline{\theta}_n = \frac{1}{D}$ and $S_n = \frac{1}{D}(\lim_{\theta_{1,n} \to 1} \frac{\theta_{1,n}}{\overline{\theta}_n} \log_2 \frac{\theta_{1,n}}{\overline{\theta}_n} + \sum_{m=2}^{D} \lim_{\theta_{m,n} \to 0} \frac{\theta_{m,n}}{\overline{\theta}_n} \log_2 \frac{\theta_{m,n}}{\overline{\theta}_n} = \frac{1}{D}(D \log_2 D + \sum_{m=2}^{D} 0) = \log_2 D$. Note that this is both the maximum value of the function and the point where a concept is as specific as it could possibly be. Thus, the concept specificity is a useful measure of how specific concepts are and is the primary measure of Specificity.

5.1.2 Experimental Results

The results of measuring Specificity at various levels of the hierarchy can be seen in Figure 4. The concepts at level 2 are clearly more specific than the concepts at level 1. Concepts at deeper levels also tend to be more specific, although the precise trend is less well defined due to less data. It seems evident that the increase in specificity is notably more gradual deeper in the hierarchy, however. Overall, it is safe to conclude that going deeper in the hierarchy does indeed result in more specific documents, although they never get significantly more specific past level 2.

The results of measuring Diversity at various levels of the hierarchy can be seen in Figure 5. For each branch at each level, the average document divergence is computed against every sibling branch. Note that at every level, the sibling branches show an increase in divergence. This indicates that different branches tend to capture different

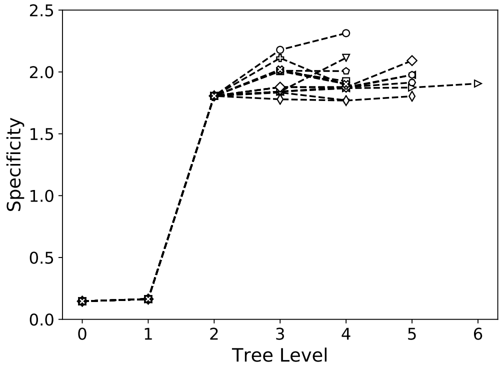


Figure 4: Specificity of concepts, i.e. S_n where n indexes a concept. The figure only shows branches whose length is no less than 5 levels.

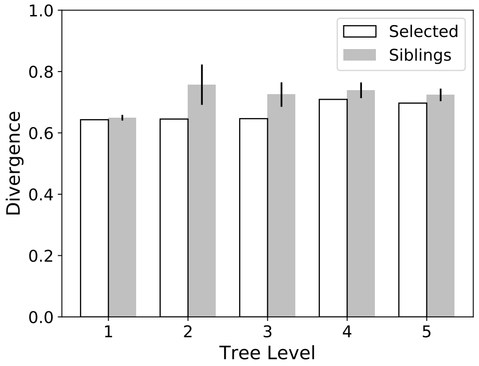


Figure 5: Average document divergence within a subset of documents selected using a concept, and average divergences and standard errors to the subset of documents from using sibling concepts.

sets of documents.

Between these two results, it can be concluded that the hierarchy does indeed organize concepts from general to specific, affirmatively answering evaluation criteria 1. The data in Figure 4 and Figure 5 suggests that the hierarchy will be effective for finding more specific or more general tags from a known tag and also be effective at finding related sibling tags, but it will be most effective at level 2. This answers evaluation criteria 2. Finally, it can be inferred that the hierarchy will be useful for an exploratory search because of both the general to specific organization and because different branches lead to different sets of documents as per Figure 5.

5.2 Tag Prediction

The L2H model was used to predict tags for a set of 23,000 unseen questions. For each question, the number of true tags within the L2H model's top N tags is used to measure the accuracy of the model for various values of N. The same test was performed with a Labeled LDA model as a point of comparison. The results are plotted in Figure 6.

In both models, accuracy increases with respect to N. This is expected: with the definition of accuracy used, it is impossible for accuracy to decrease with respect to N. Assume for a given question, it has y true tags, x of which are predicted by a model at threshold N. The accuracy of the model at threshold N will be $A(x, y) = \frac{x}{y}$. At threshold $N + 1$, there will be exactly 1 new tag predicted by the model. If this tag is not one of the true tags, then the thresholds x and y will remain unchanged, and by proxy the accuracy will be the same. If the tag is one of the true tags, then x will increase to $x + 1$, making the new accuracy $\frac{x+1}{y}$ which is strictly larger than $\frac{x}{y}$. Therefore, increasing N will always result in greater or equal accuracy for the model.

5.2.1 Plausibility

Several observations can be made about the accuracy in Figure 6. For tight thresholds (e.g. $N < 10$), accuracy is close to 45%. Much higher accuracy would be expected at these thresholds if the model was perfectly emulating human tagging behavior. However, the objective of tag predictions is not to accurately emulate human tagging, but to suggest relevant tags that may be overlooked by human taggers. In that context, suboptimal accuracy can stem both from inaccurate predictions and from useful predictions that are typically overlooked by human taggers. The concept that a prediction is useful independent of whether it is typically captured by human labeled datasets is referred to as a prediction's "plausibility."

5.2.1.1 Concept

Without a rigorous method for classifying a prediction as either inaccurate or useful, the plausibility of the predictions cannot be directly measured. An indirect method of measuring plausibility can be performed if the probability of a given prediction from the model being correct can be known a priori.

To illustrate why, assume for a moment that there is a such a way to know the a priori probability that a given prediction from the model will agree with a human tag. Consider the case where there is a strong a priori probability of a prediction agreeing with a human tag, and yet the prediction turns out to not match any human tags. In such a case, it is much more likely for that prediction to be plausible than for the prediction to be entirely inaccurate. This is so because the predictions from the model are known to be consistent with some logic (that is, whatever math and logic govern the model's behavior) that largely agrees with human reasoning for tag selection. If the logic did not largely agree with human reasoning, then it would not be able to have a strong a priori probability of a prediction agreeing with a human tag. Furthermore, the model is known to be operating in "good faith." That is, it is not doing anything to artificially reduce the quality of its predictions, such as randomly returning known bad predictions. From this, it can be inferred that the prediction is most likely plausible because it was generated in good faith by a process that largely agrees with human reasoning.

Consider the alternative case, where there is a very weak a priori probability of a prediction agreeing with a human tag and the prediction does not match any human tags. In such a case, it is much more likely for that prediction to be inaccurate. This is so because by having such a weak a priori probability, it indicates that the model does not usually agree with human reasoning in that case. As such, an incorrect prediction is most likely completely divergent from human consideration, and therefore is inaccurate.

From these two cases, it can be inferred that the likelihood of a prediction being plausible instead of inaccurate is directly proportional to the a priori probability of the prediction agreeing with a human tag. Therefore, one possible way of indirectly measuring plausibility is by measuring this a priori probability.

5.2.1.2 Derivation

It is possible to measure the probability of the model's prediction at rank N agreeing with a human tag by examining the slope of accuracy with respect to N. The proof of such is as follows:

For a given model, the probability that a prediction at rank N is a human prediction can be denoted as $P(N)$. By the frequentist definition of probability, $P(N) = \frac{a}{a+b}$, where a is the number of instances where a prediction at rank N truly agrees with a human prediction, and b is the number of instances where a prediction at rank N does not agree with any human prediction. Additionally, note that $a + b = T$, where T is the total number of predictions made at rank N, and T is a constant.

The accuracy, as defined earlier, is $A(x, y) = \frac{x}{y}$, where x is the number of tags predicted by the model within the top N that agree with human predictions for a given question, and y is the total number of tags given by humans for a given question. The slope of accuracy with respect to N, then, is $m(A, N) = \frac{A(x_2,y_2)-A(x_1,y_1)}{N_2-N_1} = \frac{\frac{x_2}{y_2}-\frac{x_1}{y_1}}{N_2-N_1}$. As y is constant, and $N_2 = N_1 + 1$, this simplifies to $\frac{x_2-x_1}{y}$. Considering the definition of x, $x_2 - x_1$, once summed over all questions, will equal the number of tags predicted by the model at exactly rank N that agree with human tags. This is the definition of a stated earlier. Thus, the slope of accuracy with respect to N is $m(A, N) = \frac{a}{\sum_{i=0}^{k} y_i}$, where k is the total number of questions. Note that $C = \sum_{i=0}^{k} y_i$ is a constant, as all y values are constant, as is k. Therefore the slope of accuracy with respect to N can be further simplified $m(A, N) = \frac{a}{C}$ where C is a constant. Finally, $P(N) = \frac{a}{a+b} = \frac{Cm(A,N)}{a+b} = \frac{Cm(A,N)}{T} = m(A, N)\frac{C}{T}$, and $\frac{C}{T}$ is a constant as it is a fraction of two constants. Therefore, for a given model and given N, $m(A, N)$ is directly proportional to the probability that predictions at rank N agree with human predictions.

5.2.2 Experimental Results

Figure 6 can offer insights about the plausibility of the predictions for L2H and LLDA. In the case of LLDA, the slope is largely flat after a small N. Therefore, it can be inferred that the predictions from LLDA are likely to have low plausibility. Conversely, L2H maintains a notable slope throughout the entire range examined, suggesting that the predictions across the entire range are relatively plausible compared to LLDA. Additionally, L2H offers better accuracy than LLDA at all examined thresholds, suggesting that L2H is an all around superior model for the task of tag recommendation.

Figure 7 shows how accuracy improves with respect to document length. The accuracy of tags increases with respect to questions length for both models. This is a logical result for topic models. Broadly speaking, each topic in a topic model captures the statistical frequencies of words in a question that would be represented by that topic. With more words, the statistical frequencies will be more accurate, and therefore more accurately portray which topics should be relevant. L2H outperforms LLDA at all examined document lengths.

The accuracy in Figure 7 is based on the top 5 tags. The results are potentially misleading if LLDA can outperform L2H at different levels of top N. Figure 7 shows that this is untrue in the general case, as L2H outperforms LLDA at all thresholds on average. However, it is still possible for LLDA to outperform L2H at specific thresholds with documents of a specific length. This possibility was not examined. It is believed that with the existing data, any such result is more likely to be statistical noise than anything meaningful, and therefore irrelevant. Nevertheless, if more data were to show that LLDA does indeed outperform L2H in a specific case (e.g. if LLDA outperformed LLDA in the case of top 10 tags with documents of length 120), this could offer a hint at a potential weakness in the L2H model. This may be a potential direction for future research. However, it is not considered a threat to validity because even in such a case, it remains true that L2H outperforms LLDA on average.

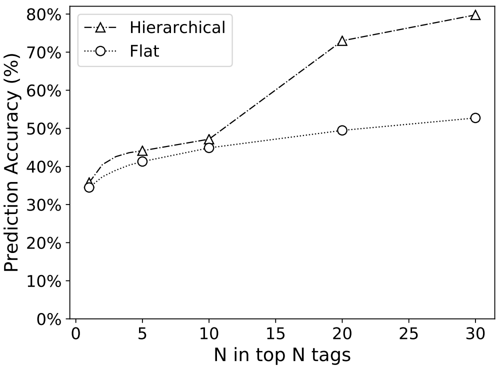


Figure 6: Tag prediction accuracy using a flat model and that using a concept hierarchy. We consider a correct prediction when the tags of an unseen question appears in N most significant topics/concepts/tags (or top N tags).

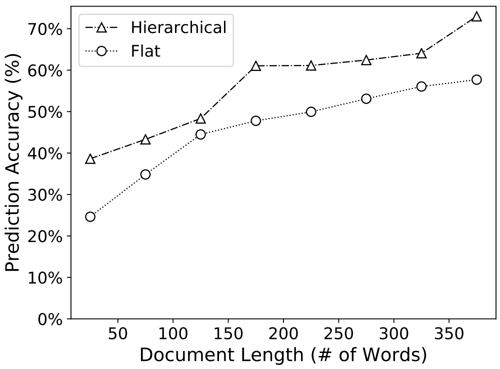


Figure 7: Tag prediction accuracy using a flat model and that using a concept hierarchy. The accuracies shown are for top 5 tags (N=5). Each accuracy measure is estimated for unseen questions whose length has an upper bound of a specific length, e.g., $(0-50]$, $(50-100]$, etc.

Overall, the results show that L2H performs effectively at tag prediction compared to the baseline. The level of accuracy and plausibility shown in the results is sufficient to give an affirmative answer to evaluation criteria 3, as L2H is indeed predicting similar tags to humans for unseen questions, even if the tags are not precisely the same. The plausibility also suggests that L2H's recommendations should be effective even when they don't precisely match human tags, answering evaluation criteria 4.

Note that in highly accurate tag recommendation methods, individual models such as L2H or LLDA are often not used directly. Instead, methods like LLDA are often used in an ensemble of classifiers, such as the ensemble EnTagRec [47]. Based on the results here, adding a model similar to L2H to such an ensemble should offer both improvements to prediction accuracy and improvements to the plausibility of results that do not match labels.

5.3 Tag Synonym Identification

In the methodology section, two potential methods of identifying tag synonyms were selected. Method 1 simply uses the Jensen-Shannon divergence between two topics and says they are synonyms if below a certain threshold. More specifically, two tags with topics ϕ_0 and ϕ_1 respectively are synonyms if and only if $JS(\phi_0, \phi_1) \leq \epsilon_1$ for some appropriate threshold ϵ_1.

Method 2 additionally requires that the two topics be near each other in the tag hierarchy to be considered synonyms. More specifically, two tags with topics ϕ_0 and ϕ_1 respectively are synonyms if and only if $\delta(\phi_0, \phi_1) \leq \epsilon_0 \wedge JS(\phi_0, \phi_1) \leq \epsilon_1$ for some appropriate thresholds ϵ_0 and ϵ_1.

To determine how effective tag synonym identification is, all $\binom{369}{2} = 67,896$ pairs of tags in the dataset are evaluated with both of these metrics. The results are compared against the human labeled tag synonym pairs, where the precision, recall, and F1 score are computed. For both methods, some threshold must be used for comparison. As the ideal threshold is unknown, a variety of thresholds are tested and compared, shown in Figure 9, Figure 8, and Figure 10.

5.3.1 Mathematical Constraints of Results

Overall, Method 2 performs better in terms of F1 and precision, while Method 1 performs better in terms of recall. As illustrated in Figure 9, reducing the maximum allowed hierarchy distance also reduces recall across all possible choices for the threshold. To some extent, this result is intuitive from the mathematics.

Recall is defined as $R = \frac{TP}{TP+FN}$, where TP is the number of true positives and FN is the number of true negatives. According to the equation for Method 2, reducing the hierarchy distance threshold ϵ_0 can only produce more additional negative results as per the definition of a logical and. Therefore, the false negatives can only be increased and the true positives can only decrease with respect to the hierarchy distance threshold ϵ_0.

Consider the case where a single false negative is added. The recall is changed to $R_1 = \frac{TP}{TP+FN+1}$, meaning the net change in recall is $\Delta R_1 = R_1 - R_0 = \frac{TP}{TP+FN+1} - \frac{TP}{TP+FN}$. This can be simplified to $\Delta R_1 = \frac{-TP}{(TP+FN)(TP+FN+1)}$. Note that ΔR_1 is negative for all positive non-zero integer values of TP and FN. Thus, increasing the number of false negatives uniformly decreases the recall value.

Consider the case where a single true positive is removed. The recall is changed to $R_2 = \frac{TP-1}{TP+FN-1}$, meaning the net change in recall is $\Delta R_2 = R_2 - R_0 = \frac{TP-1}{TP+FN-1} - \frac{TP}{TP+FN}$. This can be simplified to $\Delta R_2 = \frac{-FN}{(TP+FN)(TP+FN-1)}$. Note that ΔR_2 is negative for all positive non-zero integer values of TP and FN. Thus, decreasing the number of true positives uniformly decreases the recall value.

Therefore, decreasing the hierarchy distance threshold will always result in a smaller or equal recall value. Figure 9 is therefore not surprising. This does not imply that Method 2 is inherently inferior however because it is possible to increase the Jensen-Shannon threshold ϵ_1 to achieve higher recall. If Method 2 can achieve better precision at a similar recall, then that would be a potential reason to prefer Method 2 in some applications.

This may raise the question of whether Method 2 could even theoretically improve on Method 1 on any of the given criteria. F1 score is computed from precision and recall; thus Method 2 can be an improvement over Method 1 on the existing criteria if and only if Method 2 can theoretically improve the precision.

Precision is defined as $P = \frac{TP}{TP+FP}$. As previously established, Method 2 cannot increase the number of true positives. If a single true positive is removed, then the precision is changed to $P_1 = \frac{TP-1}{TP+FP-1}$. Note that this equation is identical to R_2 from earlier, only with FP substituted for FN. Therefore, similar to R_2, removing a single true positive can only reduce the precision for all positive non-zero integer values of TP and FP.

According to the equation for Method 2, reducing ϵ_0 can only produce more negative results. Thus, false positives can only be decreased by Method 2. If a single false positive is removed, then the precision is changed to $P_2 = \frac{TP}{TP+FP-1}$, meaning the net change in precision is $\Delta P_2 = P_2 - P_0 = \frac{TP}{TP+FP-1} - \frac{TP}{TP+FP}$. This can be simplified to $\Delta P_2 = \frac{TP}{(TP+FP-1)(TP+FN)}$. Note that ΔP_2 is positive for all positive non-zero integer values of TP and FP. Thus, decreasing the number of false positives uniformly increases the precision. Therefore, it is theoretically possible for Method 2 to improve the precision for a given threshold, and so it is indeed possible for Method 2 to offer a better result than Method 1 on the given criteria.

5.3.2 Influence of the Hierarchy

Figure 11 shows how distant topics are from each other for various possible tree distances. As expected, topics farther apart in the hierarchy tend to also be farther apart in topic distance. The variability at each tree distance is very large however, implying that correlation between the tree distance and topic distance is not particularly strong.

In theory then, if the hierarchy offers useful information, then Method 2 should in some sense be superior to Method 1 due to incorporating more useful information in its classification. The choice of metric for overall ability is somewhat complicated by the nature of the problem. Specifically, there are significantly more non-synonym pairs than synonym pairs. With an unbalanced dataset, a typical metric such as accuracy is almost entirely useless.

The F1 score is another common metric that was not hindered by the data imbalance issue in this case. The F1 score supports the hypothesis that Method 2 offers a better overall classification ability, as demonstrated in Figure 10. If thresholds are selected to optimize F1 score, then the best thresholds are at $\epsilon_0 = 1, \epsilon_1 = 0.512$ with a score of 0.7325. In contrast, for method 1 the best thresholds are at $\epsilon_1 = 0.49$ with F1 score of 0.6967. This result supports the notion that Method 2 can achieve better precision for a given Jensen-Shannon threshold, and ultimately allows better overall performance despite relaxing said threshold.

Nevertheless, there are a variety of strengths and weaknesses to both methods. Method 1 still has a number of strengths compared to Method 2, including that is is simpler to define and configure, requires less information to run (i.e. only needs the topics, not the entire hierarchy), and it offers superior recall at every threshold. Particularly, the fact that Method 1 does not depend on the hierarchy suggests that this method could be applicable to other topic models even if those models do not generate a hierarchy like L2H. By contrast, Method 2 could only potentially apply to other models if they also generate a relevant hierarchy.

5.3.3 ROC and Precision-Recall Curves

Another way of quantifying the performance of tag synonym identification is with the Receiver Operating Characteristic curve, plotting true positive rate versus false positive rate as shown in Figure 12. This curve can be used to compare the possible performances of the L2H tag synonym identification schemes with that of a non-hierarchical model. In this case, the LLDA model is used as a flat topic model for comparison. LLDA tag synonym predictions are performed with the Jensen-Shannon distance between topics, as in to Method 1. For the L2H based methods, both Method 1 (the $\delta = \infty$ model) and Method 2 (the $\delta = 1$ model) are shown.

For all models, an ideal threshold would have a high true positive rate and a low false positive rate. In this case, true positives increase rapidly with respect to false positives, suggesting that such a threshold can indeed be chosen. Furthermore, both L2H based methods appear to offer superior true positive rates compared to the LLDA model. However, this analysis can be misleading due to the extreme class imbalance in this problem.

An alternative to the ROC curve is the precision-recall curve, shown in Figure 13, which is less sensitive to data imbalance issues. As before, both L2H models are represented and compared to the LLDA baseline. In this case, an ideal threshold has both high precision and high recall. For all models, the precision remains relatively high up to a certain recall level where it drops off. The LLDA models drops off significantly sooner than either of the L2H models. Method 2 maintains better precision than Method 1 as recall increases, as has been suggested earlier.

A number of observations can be made from Figure 13. It is significant that Method 1 with the L2H model outperforms the same tag synonym identification method applied to the LLDA model. This observation suggests that topics learned in the hierarchical fashion employed by L2H are superior for synonym identification compared to the same topics learned in a flat, non-hierarchical fashion, even if the hierarchy itself is not used in making the predictions. It further suggests that if the hierarchy is used to augment the identification process, then superior results can be achieved. Note that only the most extreme instances of hierarchy distance were compared in Figure 12 and Figure 13. While this is sufficient to show the hierarchy is useful for tag synonym identification, it is entirely possible that different thresholds or more sophisticated methods of including the hierarchy in identification process could increase prediction performance further.

5.3.4 Optimal Thresholds

Overall, tag synonym identification with the outlined approaches requires careful selection of thresholds. Different thresholds can optimize different types of error rates in order to get the best performance for a desired application. Among the four metrics considered in this analysis, precision, recall, F1 score, and accuracy, the thresholds that optimize each are outlined in Table 2.

As indicated earlier, accuracy is a misleading metric to optimize for this problem due to the extreme number of non-synonymous tag pairs compared to the few number of synonymous tag pairs. The low precision at the optimal thresholds for accuracy is particularly indicative of this. Optimizing for recall can also be trivially dismissed as the resulting method is equivalent to classifying every pair as synonymous. The precision at those thresholds is also extremely low.

Among the metrics examined, then, only the thresholds that optimize precision and the thresholds that optimize F1 score would make reasonable sense in practice. Each of them could be useful with different desired use cases. For example, if the purpose is to identify a candidate set of potentially synonymous tags which will later be manually verified, then the thresholds that optimize F1 score would likely be most effective. If the thresholds that optimized precision were used instead, then fewer potentially synonymous tags would need to be rejected, but a significantly larger number of true tag synonyms would be rejected by the classifier.

Conversely, if the purpose is to automatically identify tag synonyms directly, then the thresholds that optimize precision make the most sense. If the thresholds that optimize F1 score were used instead, then more false positives will be present, increasing the burden of rejecting false synonyms and devaluing the ability of automatic tag synonym identification. The missed true synonyms are not as significant in this case, since it is assumed they are likely to be manually identified later.

5.3.5 Baseline Comparison

The TSST tag synonym recommendation method by Beyer et. al. is a potential point of comparison for the performance of the L2H based tag synonym identification schemes [48]. Unlike the LLDA and L2H models previously examined, their approach is based directly on tags themselves independent of what question content is used with the tags. For example, the names of the tags ALGORITHM and ALGORITHMS differ only in plurality. Therefore, based on tag name alone, it is likely that they are synonyms. They present a total of nine such strategies generally aimed at identifying tags that have the same name written in different ways. The output of their model is a ranked list of

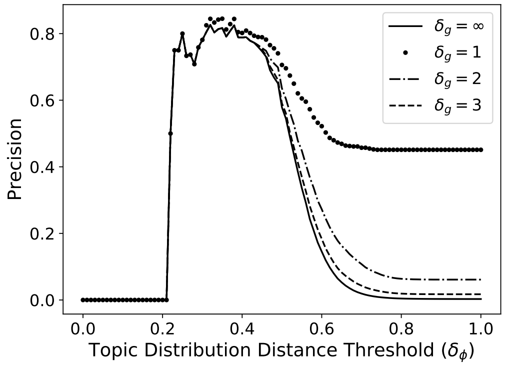


Figure 8: Precision of tag synonym identification versus topic distribution distance where the precision is defined as $\frac{TP}{TP+FP}$.

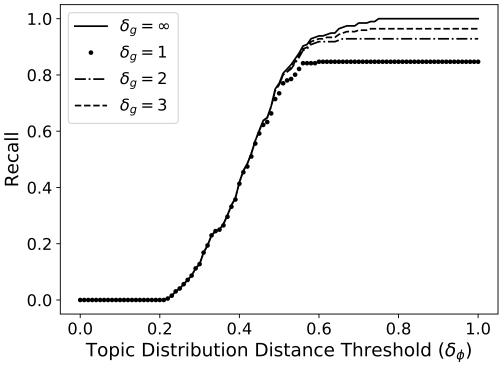


Figure 9: Recall of tag synonym identification versus topic distribution distance where the recall is defined as $\frac{TP}{TP+FN}$.

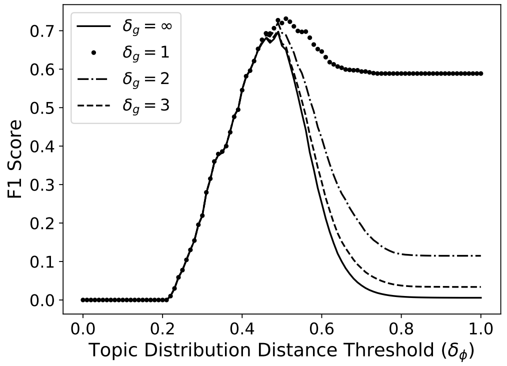


Figure 10: F1 score of tag synonym identification versus topic distribution distance where the F1 score is defined as $\frac{2TP}{2TP+FN+FN}$.

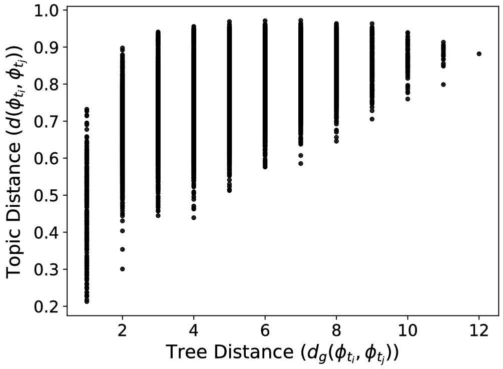


Figure 11: Topic distribution distance versus tree distance.

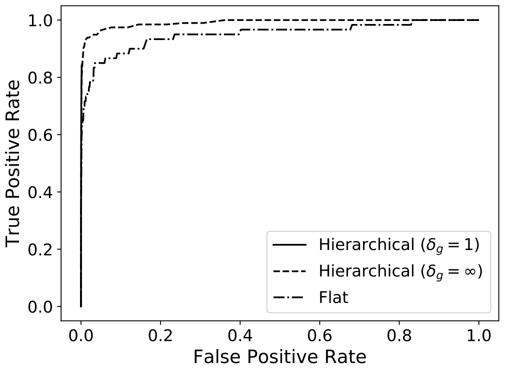


Figure 12: The Receiver Operating Characteristics (ROC) for tag-synonym identification using hierarchical model (L2H) and flat model (LLDA), respectively.

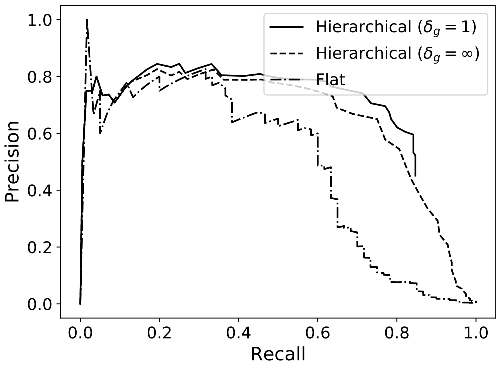


Figure 13: The Precision-Recall Characteristics (PRC) for tag-synonym identification using hierarchical model (L2H) and flat model (LLDA), respectively.

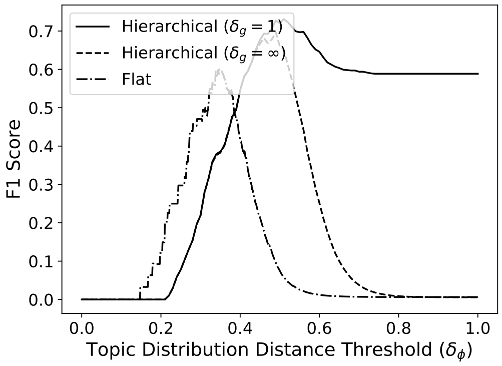


Figure 14: The F1 Scores for tag-synonym identification using hierarchical model (L2H) and flat model (LLDA), respectively.

possible synonyms for each given tag. They evaluate their model based on whether the correct synonym is found within a certain number of rank positions from the top of the list.

In their results, the correct synonym could be found within the top 15 suggestions in 74.9% of instances. The correct synonym is the top suggestion in 45.9% of instances. When tested on new synonyms not part of the known set, the synonyms are the top suggestion in 20.0% of instances. As this method of evaluation is different from the evaluation method used for L2H synonyms, these results are not directly comparable. Nevertheless, some inferences can be made about how the L2H synonyms perform relative to TSST.

As discussed earlier, when using the L2H synonym model for identifying a candidate set of potentially synonymous tags, the thresholds that optimize F1 score will be most appropriate. Therefore, the version of the model that optimizes F1 score will be the focus for comparison. In that version of the L2H model, 88.47% accuracy is achieved for identifying whether tag synonym pairs are synonymous. This figure compares favorably with the TSST model even within the TSST's top 15, with the caveat that TSST is attempting to generate a list of synonyms for a given tag whereas the L2H model is attempting to determine whether two specific tags are synonymous.

Despite the difference in evaluation methodology, it appears that the L2H model performs at least comparably to the TSST model. This conclusion is based on the superior accuracy figures compared with those reported for the TSST model, the fact that the superior figures are achieved without caveats such as within a set of 15 tags, and that the bounds of tag similarity afforded in the L2H model are typically much smaller than a set of 15 tags.

There are some threats to the validity of this analysis. Although the L2H model performs well in pairwise comparisons, it is possible that ranking tags by similarity may result in the truly synonymous tag being ranked below multiple other tags, which would harm performance if evaluated similarly to the TSST evaluation. This is mathematically unlikely given the aforementioned tight bounds of tag similarity, however. Conversely, there is a possibility that the TSST model may perform unexpectedly well if modified to perform a pairwise synonym test similar to the L2H model evaluation. Given the nature of the TSST model, it is unknown what performance would result or even if such an evaluation is feasible. Either possible threat, if true, would undermine this analysis and suggest inferior performance of the L2H model for tag synonym identification.

Overall, the proposed methods show that L2H can effectively identify tag synonyms on at least a comparable level to existing methods, if not better, answering evaluation criteria 5. While the topics alone are effective at this task, using the hierarchy for predictions improves performance. A notable additional result is that the topics alone are more effective than the same topics trained with a non-hierarchical model, even when the hierarchy is not directly used for predictions.

Table 2.: Example results of tag synonym identification

Distribution Threshold (δ_ϕ)	Graph Threshold (δ_g)	Optimal Metric	Performance Metrics

0.35	1	Precision=84.48%	Precision=84.48%
			Recall=25.00%
			Accuracy=62.49%
			F1 Score=38.58%

0.75	∞	Recall=100.00%	Precision=0.79%
			Recall=100.00%
			Accuracy=81.91%
			F1 Score=1.58%

0.60	∞	Accuracy=96.14%	Precision=14.55%
			Recall=84.69%
			Accuracy=96.14%
			F1 Score=64.59%

0.51	1	F1 Score=73.12%	Precision=69.58%
			Recall=77.04%
			Accuracy=88.47%
			F1 Score=73.12%

CHAPTER 6

CONCLUSION

This thesis examines the effectiveness of L2H for improving tagging quality on Stack Overflow. Existing research suggests that higher quality tagging could result in an overall improve Q/A experience for the site [5]. L2H is a particular interesting approach to this issue because it offers two different potential solutions to the problem. Firstly, L2H offers the ability to recommend relevant tags when a question is being asked. Secondly, L2H offers the ability to organize the tags in a hierarchy from general to specific which can assist in exploratory searching.

Five criteria were defined and investigated in detail to evaluate how well L2H realize its apparent merits.

1. Does L2H produce a hierarchy organizing tags from general to specific? Based on the Specificity metric used, the hierarchy does indeed organize tags from general to specific. The effect is strongest at level 2, with deeper levels having smaller differences in specificity.

2. How effective is the hierarchy in finding related tags from known tags? Based on the Diversity metric and the Specificity metric, related tags tend to be placed near potentially known tags. This is true in both the directions of parent to child and vice versa, as well as between sibling branches. From this, it can be inferred that the hierarchy is effective at identifying related tags based on known tags.

3. Does L2H predict similar tags to human chosen tags on unseen questions? Throughout various experiments, L2H achieved higher accuracy at predicting human tags compared to the baseline. From this, it can be said that L2H does predict similar tags to humans even on unseen questions.

4. How effective is L2H at recommending tags for human tagging? Based on the plausibility metric defined, the tags recommended by L2H which do not exactly match human tags tend to be more plausible than baseline models. In effect, this suggests that L2H will be effective at recommending tags for human tagging.

5. How effective is L2H at predicting tag synonyms? Between the methods of identifying tag synonyms examined, both were shown to be effective, although the method that included the hierarchy was more effective overall.

Overall, L2H performed successfully at all criteria, suggesting that L2H is a potentially effective solution for improving tagging quality on Stack Overflow. It is noteworthy that although L2H achieves the criteria, usually it comes with caveats. For example, while L2H does organize tags from general to specific, the most significant difference is at level 2. This could be due to the limited sample size preventing quality organization at deeper levels in the hierarchy. It may also be a hint at some unknown limitations of the model. This could be a potential area for future research.

The hierarchy was deemed effective for identifying related, but unknown tags because it placed related tags near each other. This is believed to be an effective evaluation because it follows from results in recent research on information retrieval. A potential future research direction is obtaining further validation could by performing a human trial confirming the hierarchy is useful in various use cases.

L2H was deemed effective at recommending tags because it achieved higher accuracy and plausibility than a reasonably chosen baseline. This leaves ambiguity in precisely how effective L2H is, as results were relative to an LLDA model. Comparing L2H with more models could offer more insight into how L2H is performing relative to other alternatives. A human evaluation where humans rate the quality of various suggestions could offer additional insight into how the model performs. Both are potential directions for future research. Additionally, it may be insightful to examine how L2H performs as part of an ensemble of various models for tag recommendations.

Although two methods of identifying tag synonyms were examined and deemed successful, two more sophisticated methods were left untouched. These include training a classifier with topic and hierarchy distance as features as well as using the internal logic of L2H to detect redundant topics. Additionally, if the evolution of tag usage over time could be included in a synonym identification scheme, then even higher accuracy could be achieved in this area. All of these are potential directions for future work.

Other future research directions may include applying L2H to sources outside of Stack Overflow. In doing so, relevant Stack Overflow articles could be found for documentation and tutorials. On a higher level, automatically tagging software text could start building an effective search method for untagged resources.

REFERENCES

[1] Janice Singer, Timothy Lethbridge, Norman Vinson, and Nicolas Anquetil. "An Examination of Software Engineering Work Practices". In: *CASCON First Decade High Impact Papers*. Riverton, NJ, USA: IBM Corp., 2010, pp. 174–188.

[2] C. Treude, O. Barzilay, and M. Storey. "How do programmers ask and answer questions on the web?: NIER track". In: *2011 33rd International Conference on Software Engineering (ICSE)*. 2011, pp. 804–807. DOI: 10.1145/1985793.1985907.

[3] Stack Exchange, Inc. *Tags*. Available: https://stackoverflow.com/tags, retrieved on November 15, 2018, 2018.

[4] Jaime Teevan, Christine Alvarado, Mark S. Ackerman, and David R. Karger. "The Perfect Search Engine is Not Enough: A Study of Orienteering Behavior in Directed Search". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '04. Vienna, Austria: ACM, 2004, pp. 415–422. ISBN: 1-58113-702-8. DOI: 10.1145/985692.985745. URL: http://doi.acm.org/10.1145/985692.985745.

[5] M. Asaduzzaman, A. S. Mashiyat, C. K. Roy, and K. A. Schneider. "Answering questions about unanswered questions of Stack Overflow". In: *2013 10th Working Conference on Mining Software Repositories (MSR)*. 2013, pp. 97–100. DOI: 10.1109/MSR.2013.6624015.

[6] Viet-An Nguyen, Jordan Boyd-Graber, Philip Resnik, and Jonathan Chang. "Learning a Concept Hierarchy from Multi-labeled Documents". In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14. Montreal, Canada: MIT Press, 2014, pp. 3671–3679. URL: http://dl.acm.org/citation.cfm?id=2969033.2969236.

[7] Hui Chen, John Coogle, and Kostadin Damevski. “Modeling Stack Overflow Tags and Topics as a Hierarchy of Concepts”. In: *Journal of Systems and Software* Under Review.tbd (2019), tbd.

[8] Lena Mamykina, Bella Manoim, Manas Mittal, George Hripcsak, and Björn Hartmann. “Design Lessons from the Fastest Q&#38;a Site in the West”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’11. Vancouver, BC, Canada: ACM, 2011, pp. 2857–2866. ISBN: 978-1-4503-0228-9. DOI: `10.1145/1978942.1979366`. URL: `http://doi.acm.org/10.1145/1978942.1979366`.

[9] Ivan Srba and Maria Bielikova. “Why Is Stack Overflow Failing? Preserving Sustainability in Community Question Answering”. In: 33 (July 2016), pp. 80–89.

[10] Stack Exchange, Inc. *Why does Haskell use Mergesort instead of Quicksort?* Available: `https://stackoverflow.com/questions/52237695/why-does-haskell-use-mergesort-instead-of-quicksort`, retrieved on November 15, 2018, 2018.

[11] Stack Exchange, Inc. *How to sort a list by a private field?* Available: `https://stackoverflow.com/questions/52149721/how-to-sort-a-list-by-a-private-field`, retrieved on November 15, 2018, 2018.

[12] Stack Exchange, Inc. *Is main a valid Java identifier?* Available: `https://stackoverflow.com/questions/52264638/is-main-a-valid-java-identifier`, retrieved on November 15, 2018, 2018.

[13] Stack Exchange, Inc. *Netflix Zuul AWS Integration*. Available: `https://stackoverflow.com/questions/37120542/netflix-zuul-aws-integration`, retrieved on November 15, 2018, 2018.

[14] Stack Exchange, Inc. *How do I ask a good question?* Available: `https://stackoverflow.com/help/how-to-ask`, retrieved on November 15, 2018, 2018.

[15] Stack Exchange, Inc. *How long should we wait for a poster to clarify a question before closing?* Available: `https://meta.stackoverflow.com/questions/260263/how-long-should-we-wait-for-a-poster-to-clarify-a-question-before-closing`, retrieved on November 15, 2018, 2018.

[16] S. Kairam, N. H. Riche, S. Drucker, R. Fernandez, and J. Heer. “Refinery: Visual Exploration of Large, Heterogeneous Networks through Associative Browsing”. In: *Computer Graphics Forum* 34.3 (), pp. 301–310. DOI: `10.1111/cgf.12642`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12642`. URL: `https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12642`.

[17] Kumaripaba Athukorala, bullet Antti, Oulasvirta bullet, Dorota Głowacka, bullet Jilles, Vreeken bullet, and Giulio Jacucci. *Narrow or Broad? Estimating Subjective Specificity in Exploratory Search.* Nov. 2014. DOI: `10.1145/2661829.2661904`.

[18] Teresa Gonçalves and Paulo Quaresma. *Evaluating preprocessing techniques in a text classification problem.*

[19] Avigit K. Saha, Ripon K. Sahay, and Kevin A. Schneider. “A discriminative model approach for suggesting tags automatically for Stack Overflow questions”. In: *2013 10th Working Conference on Mining Software Repositories (MSR).* 2013, pp. 73–76. DOI: `10.1109/MSR.2013.6624009`.

[20] Xiaomin Fang, Rong Pan, Guoxiang Cao, Xiuqiang He, and Wenyuan Dai. “Personalized Tag Recommendation Through Nonlinear Tensor Factorization Using Gaussian Kernel”. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence.* AAAI’15. Austin, Texas: AAAI Press, 2015, pp. 439–445.

ISBN: 0-262-51129-0. URL: http://dl.acm.org/citation.cfm?id=2887007.2887069.

[21] Glenn Boudaer and Johan Loeckx. "Enriching Topic Modelling with Users' Histories for Improving Tag Prediction in Q&#38;A Systems". In: *Proceedings of the 25th International Conference Companion on World Wide Web*. WWW '16 Companion. Montr&#233;al, Qu&#233;bec, Canada: International World Wide Web Conferences Steering Committee, 2016, pp. 669–672. ISBN: 978-1-4503-4144-8. DOI: 10.1145/2872518.2890566. URL: https://doi.org/10.1145/2872518.2890566.

[22] Yong Wu, Yuan Yao, Feng Xu, Hanghang Tong, and Jian Lu. "Tag2Word: Using Tags to Generate Words for Content Based Tag Recommendation". In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. CIKM '16. Indianapolis, Indiana, USA: ACM, 2016, pp. 2287–2292. ISBN: 978-1-4503-4073-1. DOI: 10.1145/2983323.2983682. URL: http://doi.acm.org/10.1145/2983323.2983682.

[23] Michael Steinbach, George Karypis, Vipin Kumar, et al. "A comparison of document clustering techniques". In: *KDD workshop on text mining*. Vol. 400. 1. Boston. 2000, pp. 525–526.

[24] Rui Xu and Donald Wunsch II. "Survey of clustering algorithms". In: *IEEE Transactions on Neural Networks* 16.3 (2005), pp. 645–678. ISSN: 1045-9227. DOI: 10.1109/TNN.2005.845141.

[25] Eduardo C. Campos, Lucas B. L. de Souza, and Marcelo de A. Maia. "Searching crowd knowledge to recommend solutions for API usage tasks". In: *Journal of Software: Evolution and Process* 28.10 (2016). JSME-14-0119.R2, pp. 863–892. ISSN: 2047-7481. DOI: 10.1002/smr.1800. URL: http://dx.doi.org/10.1002/smr.1800.

[26] Siddharth Subramanian, Laura Inozemtseva, and Reid Holmes. “Live API Documentation”. In: *Proceedings of the 36th International Conference on Software Engineering*. ICSE 2014. Hyderabad, India: ACM, 2014, pp. 643–652. ISBN: 978-1-4503-2756-5. DOI: 10.1145/2568225.2568313. URL: http://doi.acm.org/10.1145/2568225.2568313.

[27] Stack Exchange, Inc. *Call and receive output from Python script in Java.* Available: https://stackoverflow.com/questions/10097491/call-and-receive-output-from-python-script-in-java, retrieved on November 15, 2018, 2018.

[28] Adler J. Perotte, Frank Wood, Noemie Elhadad, and Nicholas Bartlett. “Hierarchically Supervised Latent Dirichlet Allocation”. In: *Advances in Neural Information Processing Systems 24*. Ed. by J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger. Curran Associates, Inc., 2011, pp. 2609–2617. URL: http://papers.nips.cc/paper/4313-hierarchically-supervised-latent-dirichlet-allocation.pdf.

[29] Yves Petinot, Kathleen McKeown, and Kapil Thadani. “A Hierarchical Model of Web Summaries”. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*. HLT ’11. Portland, Oregon: Association for Computational Linguistics, 2011, pp. 670–675. ISBN: 978-1-932432-88-6. URL: http://dl.acm.org/citation.cfm?id=2002736.2002866.

[30] Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. “An Introduction to MCMC for Machine Learning”. In: *Machine Learning* 50.1 (2003), pp. 5–43. ISSN: 1573-0565. DOI: 10.1023/A:1020281327116. URL: https://doi.org/10.1023/A:1020281327116.

[31] Jack Edmonds. “Optimum Branching”. In: *Journal of Research of the National Bureau of Standards* 71B.4 (1967), 233–240.

[32] Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D. Manning. *Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora.*

[33] Amr Ahmed, Liangjie Hong, and Alexander J Smola. "The Nested Chinese Restaurant Franchise Process: User Tracking and Document Modeling". In: *International Conference on Machine Learning (ICML).* 2013.

[34] W. K. Hastings. "Monte Carlo sampling methods using Markov chains and their applications". In: *Biometrika* 57.1 (1970), pp. 97–109. DOI: `10.1093/biomet/57.1.97`. eprint: `/oup/backfile/content_public/journal/biomet/57/1/10.1093_biomet_57.1.97/1/57-1-97.pdf`. URL: `http://dx.doi.org/10.1093/biomet/57.1.97`.

[35] Chatterjee Preetha, Manziba Akanda Nishi, Kostadin Damevski, Vinay Augustine, Lori Pollock, and Nicholas A. Kraft. "What Information about Code Snippets Is Available in Different Software-Related Documents? An Exploratory Study". In: *IEEE 24th International Conference on Software Analysis, Evolution, and Reengineering* (2017).

[36] Dave Binkley, Dawn Lawrie, and Christopher Morrell. "The need for software specific natural language techniques". In: *Empirical Software Engineering* 23.4 (2018), pp. 2398–2425. ISSN: 1573-7616. DOI: `10.1007/s10664-017-9566-5`. URL: `https://doi.org/10.1007/s10664-017-9566-5`.

[37] P. C. Rigby and M. P. Robillard. "Discovering essential code elements in informal documentation". In: *2013 35th International Conference on Software Engineering (ICSE).* 2013, pp. 832–841. DOI: `10.1109/ICSE.2013.6606629`.

[38] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis.* Vol. 2. CRC press Boca Raton, FL, 2014.

[39] Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L. Boyd-graber, and David M. Blei. "Reading Tea Leaves: How Humans Interpret Topic Models". In: *Advances in Neural Information Processing Systems 22*. Ed. by Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta. Curran Associates, Inc., 2009, pp. 288–296. URL: `http://papers.nips.cc/paper/3700-reading-tea-leaves-how-humans-interpret-topic-models.pdf`.

[40] Keith Stevens, Philip Kegelmeyer, David Andrzejewski, and David Buttler. "Exploring Topic Coherence over Many Models and Many Topics". In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. EMNLP-CoNLL '12. Jeju Island, Korea: Association for Computational Linguistics, 2012, pp. 952–961. URL: `http://dl.acm.org/citation.cfm?id=2390948.2391052`.

[41] Qiaozhu Mei, Xuehua Shen, and ChengXiang Zhai. "Automatic Labeling of Multinomial Topic Models". In: *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '07. San Jose, California, USA: ACM, 2007, pp. 490–499. ISBN: 978-1-59593-609-7. DOI: `10.1145/1281192.1281246`. URL: `http://doi.acm.org/10.1145/1281192.1281246`.

[42] Jey Han Lau, Karl Grieser, David Newman, and Timothy Baldwin. "Automatic Labelling of Topic Models". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. HLT '11. Portland, Oregon: Association for Computational Linguistics, 2011, pp. 1536–1545. ISBN: 978-1-932432-87-9. URL: `http://dl.acm.org/citation.cfm?id=2002472.2002658`.

[43] Jianhua Lin. "Divergence measures based on the Shannon entropy". In: *IEEE Transactions on Information Theory* 37.1 (1991), pp. 145–151. ISSN: 0018-9448. DOI: `10.1109/18.61115`.

[44] Dominik M. Endres and Johannes E. Schindelin. "A new metric for probability distributions". In: *IEEE Transactions on Information Theory* 49.7 (2003), pp. 1858–1860. ISSN: 0018-9448. DOI: `10.1109/TIT.2003.813506`.

[45] Stack Exchange, Inc. *Create tag synonyms*. Available: `https://stackoverflow.com/help/privileges/suggest-tag-synonyms`, retrieved on February 15, 2018, 2018.

[46] Octavio Martínez and M Humberto Reyes-Valdés. "Defining diversity, specialization, and gene specificity in transcriptomes through information theory". In: *Proceedings of the National Academy of Sciences* 105.28 (2008), pp. 9709–9714.

[47] S. Wang, D. Lo, B. Vasilescu, and A. Serebrenik. "EnTagRec: An Enhanced Tag Recommendation System for Software Information Sites". In: *2014 IEEE International Conference on Software Maintenance and Evolution*. 2014, pp. 291–300. DOI: `10.1109/ICSME.2014.51`.

[48] Stefanie Beyer and Martin Pinzger. "Synonym Suggestion for Tags on Stack Overflow". In: *Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension*. ICPC '15. Florence, Italy: IEEE Press, 2015, pp. 94–103. URL: `http://dl.acm.org/citation.cfm?id=2820282.2820296`.