Tag Recommendations in StackOverflow

Logan Short, Christopher Wong, and David Zeng

***Abstract*—Many social information websites require users to organize content by marking user generated content with “tags”. In order for such sites to maintain their organization, this tagging process should be as accurate as possible. One way the website can facilitate accurate tagging is to recommend tags for users based on the content they generate. For our project, we will study tag recommendations in the context of the social question and answer site StackOverflow. We introduce the algorithm *NetTagCombine*, which utilizes the network properties of StackOverflow to improve upon existing machine learning methods for tag recommendation.**

1 Introduction

IN many community-based information web sites, such as StackOverflow, users contribute content in the form of questions and answers, allowing others to learn through the collaboration and contributions of the community. These web sites often rely upon tags as metadata that assists in the indexing, categorization, and search for particular content with just a few key words. Almost always, users are given the responsibility to choose tags which identify their own content. Tagging, however, can prove to be a confusing process for inexperienced users who may not be familiar with the tags available to them. In addition, general human error or malicious users could lead to improperly tagged posts that disrupt the organization of information on the website.

As such, there is a clear motivation behind implementing stronger and more accurate tag recommendation systems. The most obvious benefit stems from the fact that tags help ensure that website data is properly categorized and easily searchable by users. A tag recommendation model can improve tag accuracy and effectiveness in a number of ways. For example, new users would not have to worry as much when choosing appropriate tags for their questions, and so the act of asking questions is an easier experience and results in better tags being chosen. Furthermore, recommending tags decreases the possibility of introducing *tag synonyms* into the tagspace, as is commonly done through human error. Tag synonyms are discussed more in Section 3.2.

The development of tag recommendation systems for user created content is a relatively new field and most previous work has taken place within the last couple of years. Thus, tag recommendation is a field in which the state of the art is still being actively developed, and the most accurate methods for recommending tags have yet to be established. One way we believe that existing machine learning methods for tag recommendation can be improved upon is to utilize the graph features of the underlying network structure of these social information websites. Based on this intuition, the goal of our algorithm *NetTagCombine* is to analyze whether the underlying network structure of StackOverflow can be used to accurately recommend tags for user-produced content and to improve upon the current algorithms in this developing field.

2 Prior Work and Motivation

We first surveyed the recent literature and research in tag recommendation systems to get an idea of what improvements could be made. We discovered two recent papers, published in 2013 and 2014, that detail two algorithms, *TagCombine* and *EnTagRec*, for tag recommendation on sites like StackOverflow. The improvements we can make to these algorithms (Section 2.3), along with recent discussion about community detection in graphs (Section 2.4), give us motivation for a new, improved tag recommendation model.

2.1 *TagCombine Algorithm*

In [1], Xia et al. propose an automatic tag recommendation algorithm *TagCombine*. There are three components of *TagCombine*, each of which tries to assign the best tags to untagged objects: (1) multi-label ranking component, which predicts tags using a multi-label learning algorithm, (2) similarity based ranking component, which uses similar objects to recommend tags, and (3) tag-term based ranking component, which analyzes the historical affinity of tags to certain words in order to suggest tags. The recommendation algorithm methodically computes various weighted sums of the three components to attempt to find the best overall model. A $recall@k$ score is then calculated for each prediction model from stratified 10-fold cross validation. (The $recall@k$ metric is discussed more in Section 4.1.) The results demonstrate that *TagCombine* performs significantly better than all other cited models.

2.2 *EnTagRec Algorithm*

In [2], Wang et al. propose a tag recommendation system dubbed *EnTagRec*. The proposed *EnTagRec* computes tag probability scores using two separate methods, Bayesian Inference and Frequentist Inference, and then takes a weighted sum of the probability scores. Bayesian Inference relies on a posts textual data to compute the probability that a given tag is associated with the post. *EnTagRec* formulates posts into a bag-of-words model and then trains a Labeled Latent Dirichlet Allocation model which is used to compute tag probability scores for a post. The Frequentist Inference approach infers a set of tags after some preprocessing of a post, and then utilizes a network of tags to select additional tags that are similar to the ones in the set. The network of tags is constructed with the tags as nodes and weighted edges between two tags based on the Jaccard similarity of the posts that contain those tags. Experimental results show that *EnTagRec* performs significantly better than *TagCombine* from [1] on StackOverflow, Ask Ubuntu, and Ask Different datasets, but yields only comparable results on Freecode datasets.

2.3 *Motivation for NetTagCombine*

In [1], Xia et al. propose a recommendation system that relates the textual features of posts to tags with reasonably good results. However, one weakness of *TagCombine* is that it fails to look at the network structure of software information sites. Posts on sites like StackOverflow are ultimately connected to each other through an underlying network structure where users and tags that appear on multiple posts represent connections between said posts. In fact, the main purpose of tags is to group similar posts and create an organized structure that allows for more convenient and logical browsing of posts. Thus, it is not too farfetched to conjecture that knowledge of the networks structure could be used to enhance a tag recommendation system. In fact, in [2], Wang et al. use not only textual analysis of posts, but also basic network analysis of the tags. Although the results of *EnTagRec* outperformed those of *TagCombine*, it is difficult to quantify the improvement directly caused by the network features, as the machine learning models used by the two algorithms are also different. Our proposed algorithm, *NetTagCombine*, will use *TagCombine* as its baseline instead of *EnTagRec*, primarily because *TagCombine* has no network features to begin with. By adding network related components for tag recommendation to *TagCombine*, we can unambiguously quantify the improvement these network features have to offer by directly comparing our results to those of *TagCombine*. Furthermore, the machine learning models used in *TagCombine* are also simpler than those used in *EnTagRec*, making them more tractable to implement in the timeframe we were given.

2.4 *Prior Work in Community Detection*

In [3], McAuley and Leskovec discuss a method for automatically detecting "circles" in networks of users based on similarities in user profiles. This motivated us to consider using community detection in improving tag recommendation. Since tags serve as a method of organizing posts into topics, in a way, tags represent the topic-based communities of these websites. As such, algorithms that infer the community structure in network representations of these websites could give us a way to recommend tags to posts that belong to certain communities.

We considered multiple algorithms in the SNAP library for detecting communities, the Girvan-Newman method [4], the Clauset-Newman-Moore method [5], the clique percolation method [6], and a recent algorithm known as BIGCLAM [7]. Of these methods, we decided to use the BIGCLAM method for two reasons. The first comes from recent work in overlapping communities. In [8], Yang and Leskovec detail how intersections of communities in social networks are more often densely connected, contrary to the past belief that communities were densely connected and fewer connections existed at the intersections. BIGCLAM, also implemented by Yang and Leskovec, takes these newer findings on overlapping communities into account, whereas the older methods focus more on the older assumptions of the structure of social communities. In particular, Girvan-Newman and Clauset-Newman-Moore do not even allow for nodes to lie in multiple communities, which is definitely a drawback considering that posts can easily lie at the intersection of many topics. The second reason is that BIGCLAM is a parallelizable method that runs much faster than the other algorithms, allowing us to run the algorithm more times on our data. More on how BIGCLAM is utilized in *NetTagCombine* will be described in Section 4.

3 DATA AND NETWORK ANALYSIS

We begin by discussing our data collection and some key points in our preprocessing of the StackOverflow dataset in Sections 3.1 and 3.2. We then explore various graphs that can be constructed from the underlying StackOverflow network structure, which we will apply in our new tag recommendation model.

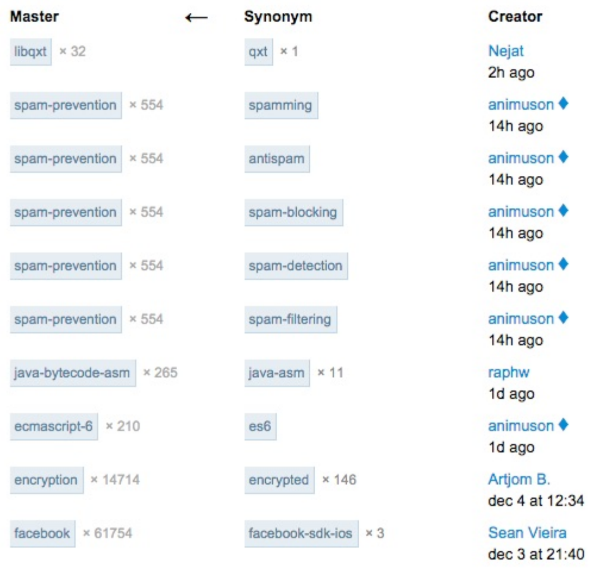
3.1 *Data Collection*

StackOverflow is a member of the StackExchange network, and all user content contributed on this network is cc-by-sa 3.0 licensed. Our data set is the September 26, 2014 snapshot for StackOverflow, downloaded from the StackExchange data dump (see [9]). The raw data set contains approximately 20 gigabytes of compressed XML files corresponding to Badges, Comments, PostHistory, PostLinks, Posts, Tags, Users, and Votes.

As in [1], we chose to look at only the first 50,000 questions posted on StackOverflow, with creation dates ranging from July 2008 to December 2008. Thus, we also stripped down our data set of users, answers, comments, and tags to only those related to 50,000 questions. By using the same data as the authors of *TagCombine*, we eliminate any discrepancies that may have arisen from doing so otherwise. We also preprocessed the text of each post, using the `snowballstemmer-1.2.0` Python package to stem word tokens. Furthermore, we removed stop words and any word tokens from our consideration set that did not appear at least 20 times, since rare words are not very helpful in generating accurate models. Finally, we considered only the tags that appeared more than 50 times among the first batch of questions, since less-common tags would not likely be recommended to a user anyways.

3.2 *Tag Synonyms*

We refer to two tags as tag synonyms if their names are different but they refer to the same concept, such as `.net-3.5` and `.net-framework-3.5`. Tag synonyms are a direct result of a question poster being given full discretion to assign tags to his or her post and to arbitrarily create new tags. This negatively impacts the strength of the tagspace, since a user searching for questions related to `.net-3.5` could completely miss the highly-related questions tagged with `.net-framework-3.5`. While we intend our tag recommendation system to help prevent future synonym groups, the currently existing groups must be addressed. Since the pruning of tag synonyms is currently done manually by volunteer contributors, there are still many synonym groups throughout the site. Figure 1 is a screenshot of the *Tag Synonyms* page of StackOverflow taken on December 9, and we can see that the maintenance of this list varies in consistency.



Master	←	Synonym	Creator
libqxt × 32		qxt × 1	Nejat 2h ago
spam-prevention × 554		spamming	animuson ♦ 14h ago
spam-prevention × 554		antispam	animuson ♦ 14h ago
spam-prevention × 554		spam-blocking	animuson ♦ 14h ago
spam-prevention × 554		spam-detection	animuson ♦ 14h ago
spam-prevention × 554		spam-filtering	animuson ♦ 14h ago
java-bytecode-asm × 265		java-asm × 11	raphw 1d ago
ecmascript-6 × 210		es6	animuson ♦ 1d ago
encryption × 14714		encrypted × 146	Artjom B. dec 4 at 12:34
facebook × 61754		facebook-sdk-ios × 3	Sean Vieira dec 3 at 21:40

Fig. 1. Screenshot of *Tag Synonyms* page.

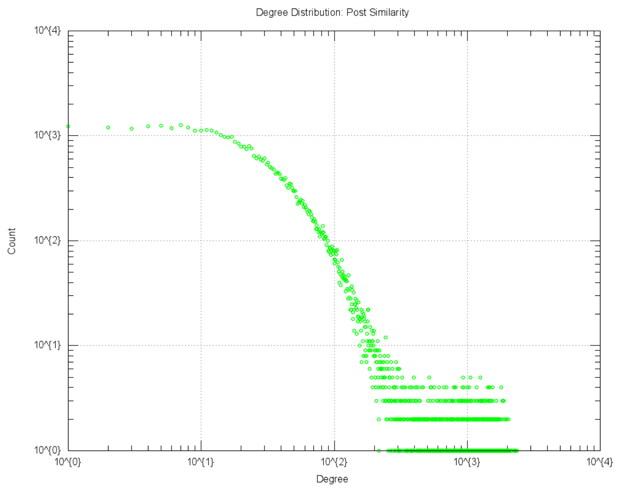
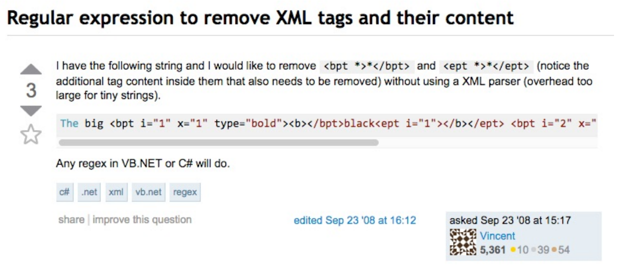


Fig. 2. Degree distribution of the Post Similarity network.



Regular expression to remove XML tags and their content

3

I have the following string and I would like to remove `<bpt *>*</bpt>` and `<ept *>*</ept>` (notice the additional tag content inside them that also needs to be removed) without using a XML parser (overhead too large for tiny strings).

```
The big <bpt i="1" x="1" type="bold"><b></bpt>black<ept i="1"></b></ept> <bpt i="2" x="
```

Any regex in VB.NET or C# will do.

c# .net xml vb.net regex

share | improve this question

edited Sep 23 '08 at 16:12

asked Sep 23 '08 at 15:17
Vincent
5,361 ●10 ●39 ●54

Fig. 3. Post #121656 from the StackOverflow website. Retrieved on 12/9/14 from http://stackoverflow.com/questions/121656.

To be able to test the effects of tag synonyms on our tag recommendation model, we prepared a separate tag list derived from our initial set of 437 tags after manually pruning for tag synonyms. Since these tags are among the more popular tags in the StackOverflow community, we were only able to reduce this new set to a size of 428 after coalescing tags such as `report` and `reporting`. We expect this pruned list of tags to moderately, but not drastically, improve our results.

3.3 Network Features

The underlying structure of the StackOverflow network is diverse and complex since users and tags can be related through various questions, answers, and comments. We picked certain relationships between objects that we deemed to likely be the most indicative of the best tags to recommend and constructed the appropriate graphs. We briefly describe them in the following sections.

3.3.1 Network Based on Post Similarity: A natural graph to consider on the StackOverflow data would be the graph in which the nodes are questions and edges connect two questions if tf-idf vectors of their textual bodies have cosine similarity above a certain threshold. This graph essentially connects posts in StackOverflow based on a measure of topical similarity. Our reasoning for choosing tf-idf vectors and cosine similarity is because *TagCombine* also uses this measure for similarity between posts, which will be discussed in Section 4.1.2. Figure 2 shows the degree distribution of such a graph when the threshold for cosine similarity is chosen to be 0.3.

We notice that this degree distribution looks much like a power law distribution. Most questions are only similar to a few other questions on StackOverflow, while a few questions are similar to many others. In particular, we can observe the kinds of questions that are similar to many others. Figure 3 shows one such question.

Notice that Figure 3 references XML parsing, C#, .NET, and regular expressions, all of which are very common topics on StackOverflow. In general, the posts that have high degrees are at the intersection of multiple popular topics. This is backed up by [8], in which we see that the intersection of communities in a network are densely connected. These well-connected posts are likely to lie at the intersection of communities in this network, which motivates us to use this graph in tag recommendation. Since tags are StackOverflows method of organizing posts into topical categories or communities, extracting communities on this graph could lead to information about what tags to recommend to new questions that are similar to posts in a given community.

3.3.2 Network Based on User Interaction: We also experimented with the following StackOverflow network. The users of StackOverflow are represented as nodes of our network, and two users u and v are linked with an edge if u answers a question posted by v such that the answer reaches a predefined threshold in positive rating. In this case, two users share similar topical interests, which lead to their interactions on StackOverflow through question and answer. Figure 4 is a plot of the degree distribution of this network when the threshold of positive rating is set to at least 2 upvotes.

The power law distribution occurs as a result of the following explanation. The users with degrees between 10 and 100 have the highest question to answer ratio, around 0.196, while the users with degrees between 1 and 10 have the lowest question to answer ratio, around 0.118. This means that a question posted on StackOverflow is likely to elicit multiple

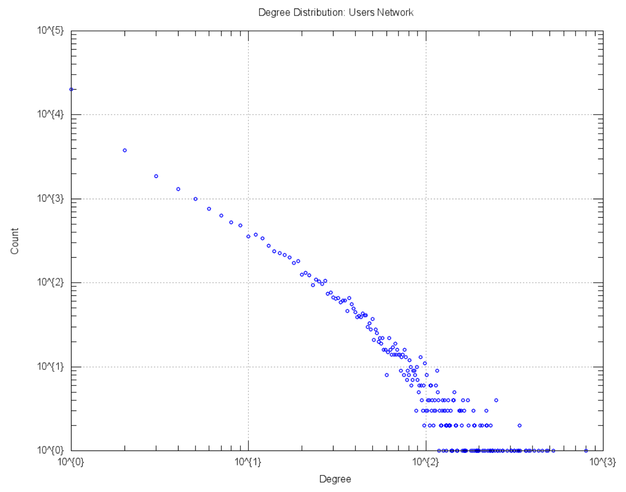


Fig. 4. Degree distribution of the User Interaction network.

answers, resulting in a medium sized degree for users that post questions. Many infrequent users that answer questions will only receive interactions with the user that posted the question leading to low degrees for these users. On the extreme end, users with degree over 100 have answered on average 148 questions each, which indicate that the tail of the distribution is populated mainly by power users or experts on StackOverflow.

The motivation for using such a graph banks on the assumption that users probably only post on a few topics in StackOverflow, and that new questions by a user are likely to share tags with a previous post by the same user. Thus, if we could organize users into communities and assign tag representatives for these communities, these tag representatives could become recommendations for questions posted by users from that community.

3.3.3 Bipartite Graph between Users and Tags: On sites such as StackOverflow, relationships exist between users and tags since users will tend to interact most with the tags they are interested in or posses the most expertise with. In order to analyze these relationships, we generated a bipartite graph where nodes on one side of the graph represented users and nodes on the opposite side of the graph represented tags. The edges of the graph were constructed and weighted to represent a users contribution and interaction with each tag. Questions, answers, and comments made by a user on a post associated with a particular tag each contributed to that users score with the tag in question. User-made questions were evaluated as the most significant form of contribution to a tag since each post is defined by the original question and thus added the most to a users tag score. Answers were evaluated as the next most significant form of contribution since answers make up the majority of the structure of a post (not including the question) and require some level of expertise with the tags associated with the post. Comments were evaluated as significantly less indicative of a contribution to a tag since they are generally not a significant contribution to the content of a post. User tag scores were then used as weights for the edges connecting each user to each tag. Tags which scored 0 points with a user did not contain an edge to that user since this meant that the user had not interacted with the tag. Figure 5 shows the degree distribution of the users in the bipartite graph.

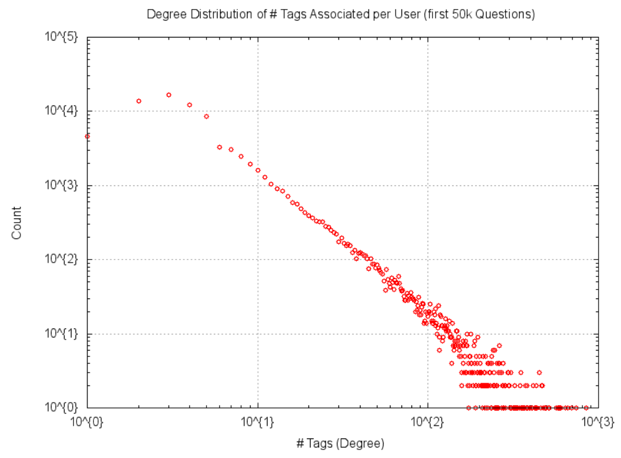


Fig. 5. Degree distribution of the bipartite graph between users and tags.

From the plot we can see that the number of tags each user was associated with followed a power law distribution. This indicates that most users interact with a very small number of tags while a select few users contribute to a large number of tags. Such a trend is consistent across many more standard examples of social networks with the vast majority of users having a very limited number of connections and only a small portion of users having a large amount of connections. Thus the degree distribution suggests that the users on StackOverflow behave in a manner similar to those of other social networks. In addition, the fact that most users are associated with a limited pool of tags suggests that recommending tags based on users associated with a post could yield accurate results. For example, if the majority of users interacting on a post all have high tag scores with the same tag, then since most users have a limited pool of associated tags, we can be fairly confident that the post in question should be assigned the shared tag.

4 Algorithm and Results

Now, we discuss our implementation of our tag recommendation models and their results on the StackOverflow data set. In Section 4.1, we implement our own version of *TagCombine* to establish a baseline with which we can compare our results. In Sections 4.2 and 4.3, we describe our improvements to propose our new algorithm, *NetTagCombine*. Finally, in Sections 4.4 and 4.5, we analyze the performance of our proposed system.

4.1 Original TagCombine Components

To start, using the procedures described in [1], we implemented the three major components of the *TagCombine* algorithm to establish a baseline for the performance of our tag recommendation system. By reproducing a working implementation of *TagCombine*, we will then be able analyze the effect of our improvements on tag recommendation accuracy.

In [1], we are introduced to the concept of the $recall@k$ metric for measuring the success of a tag recommendation model, where k is a tunable parameter that determines how many tags the model recommends for each object. Intuitively, over n objects, the $recall@k$ metric measures the average success rate in predicting correct tags for each object, where a "correct" tag is simply a tag that has been used to label that particular object by an actual user. Let R_i be the set of tags recommended for object i (so, $|R_i| = k$), and let T_i be the actual set of tags used to label object i. Then, the formula for $recall@k$ is:

$$recall@k = \frac{1}{n}\sum_{i=1}^{n}\frac{|R_i \cap T_i|}{|T_i|}.$$

We now give a quick overview of each component of *TagCombine*. A full description of each component can be found in [1].

4.1.1 Multilabel Learning Algorithm: The first component of *TagCombine* utilizes Binary Relevance to obtain the likelihood of a post to be associated with each tag given the text of the post. Given a tag set L and a question q, the Binary Relevance method will create $|L|$ datasets each of which corresponds to the classification of whether a different tag in L should be applied to q given the text in the body of q. Assuming that tags are independent from one another, *TagCombine* performs multinomial Naive Bayes on each dataset and concatenates the resulting probabilities to obtain a score vector Y whose elements Y_i correspond to the probability that the i^{th} tag is used to label the post. The values of the resulting score vector are then used to directly obtain the likelihood for each tag.

4.1.2 Similarity Based Ranking Component: The second component of *TagCombine* is based on assigning tags via similar posts. Given a new question q, *TagCombine* first finds similar posts to q using cosine similarity. First the idf of each term in our dictionary is computed. The idf of a term t is given by the formula

$$\text{idf}(t) = \frac{\#\text{ of total documents}}{\log(\#\text{ of documents containing t})}$$

Next, for all posts p, the tf-idf of a term t in a post p is computed using

$$\text{tfidf}(t,p) = \text{tf}(t,p) \times \text{idf}(t)$$

where $\text{tf}(t,p)$ is the raw frequency of the term t in post p. The tf-idf vector of a post p, $\text{tfidf}(p)$, then refers to the vector of $\text{tfidf}(t,p)$ values over all terms t. The cosine similarity between an old post p and the new question q would therefore be

$$\text{sim}(p,q) = \frac{\text{tfidf}(p) \times \text{tfidf}(q)}{\|\text{tfidf}(p)\|\|\text{tfidf}(q)\|}.$$

Using this cosine similarity formula, we can find $p_1, \ldots, p_{50}$, the 50 most similar posts to q. Let $T_1, \ldots, T_{50}$ denote the sets of tags used in each of these posts. We compute a likelihood for each tag g for question q using the following formula

$$L(g,q) = \frac{|\{i : g \in T_i\}|}{\sum_i |T_i|}.$$

Essentially, each of the top 50 posts casts a vote for each tag used in these posts. The overall likelihood of a tag g is then proportional to the number of votes received by the tag.

4.1.3 Tag-term Based Ranking Component: The third component of *TagCombine* analyzes the co-occurrence of each tag with each term in order to recommend the top tags based on a posts text. The co-occurrence, or affinity score, between a tag t and word w is defined as

$$\text{co}(t,w) = \frac{\#\text{ of posts with } t \text{ and } w}{\#\text{ of posts with } t}.$$

Then, for a given post p comprised of a set of words, the tag-term based ranking score for a tag t is calculated as

$$\text{TagAff}(p,t) = 1 - \prod_{w \in p}(1 - \text{co}(t,w)).$$

Intuitively, a tag t_0 will be recommended for a post if it shares high $\text{co}(t,w)$ affinity scores with for the words w of the post. Tag t_0 has a high affinity score with w if it had previously been used as a tag for many posts containing w.

4.1.4 Analysis of TagCombine: To establish a baseline for our tag recommendation system, we evaluated each individual component as standalone along with the entire *TagCombine* algorithm itself using 10-fold cross validation. As will be discussed more in Section 4.3, *TagCombine* methodically assigns weights α, β, and γ to the Multilabel Classifier scores, Similarity Ranking scores, and Tag Term Affinity scores, respectively. Thus, the algorithm fine-tunes these parameters to determine how much (or little) each component should be applied to produce the optimal tag recommendation system. In our runs of *TagCombine*, we found that $(\alpha, \beta, \gamma) = (0.6, 0.6, 0.8)$ produced the best $recall@k$ scores.

	$recall@5$	$recall@10$
Multilabel Classifier	0.456	0.547
Similarity Ranking	0.435	0.523
Tag Term Affinity	0.235	0.326
Our *TagCombine*	0.486	0.574
[1] Cited *TagCombine*	0.596	0.724

We can see that our standard implementation of *TagCombine* does significantly worse than the cited *TagCombine*. To remedy for this, we also calculated tag-adjusted $recall@k$ scores. From earlier, we defined T_i to be the set of all tags that are actually used to label post i. The description of $recall@k$ in [1] is ambiguous in explaining whether an element of T_i can be any tag in the StackOverflow network, as we assumed in the first table, or only the top 437 that we filtered in Section 3.1. So, we define $recallAdj@k$ scores to be calculated the same as $recall@k$ except that T_i can contain tags only from the list of the top 437. These scores will undoubtedly be higher, since our $recall@k$ scores will no longer suffer from being unable to predict tags in T_i that are outside our relevant tagspace.

	$recallAdj@5$	$recallAdj@10$
Multilabel Classifier	0.587	0.703
Similarity Ranking	0.569	0.681
Tag Term Affinity	0.304	0.420
Our *TagCombine*	0.617	0.748
Cited *TagCombine*	0.596	0.724

Our $recallAdj@k$ scores are much closer to those from the original *TagCombine* algorithm, perhaps suggesting that [1] also modified their set of tags attached to each post. However, as a reminder, we are only interested in how applying the underlying StackOverflow network can potentially improve upon our tag recommendation model, and so the interesting results will be the difference between the $recall@k$ (or $recallAdj@k$) scores of our new model and those of our baseline. Our method of calculating scores and the actual score values do not matter as long as they are consistent.

Clearly, the composite *TagCombine* score is not a linear combination of its individual components. So, before continuing on to improve *TagCombine*, we also analyzed the contribution of each individual component to the *TagCombine* scores, using $recall@5$ as an example. To do so, we held two parameters constant at their optimal values while varying the third. The Multilabel Classifier is weighted by α, Similarity Ranking is weighted by β, and Tag Affinity is weighted by γ, so to test the significance of the Similarity Ranking component, for example, we held α and γ constant while varying β. Figure 6, Figure 7, and Figure 8 plot α, β, and γ, respectively.

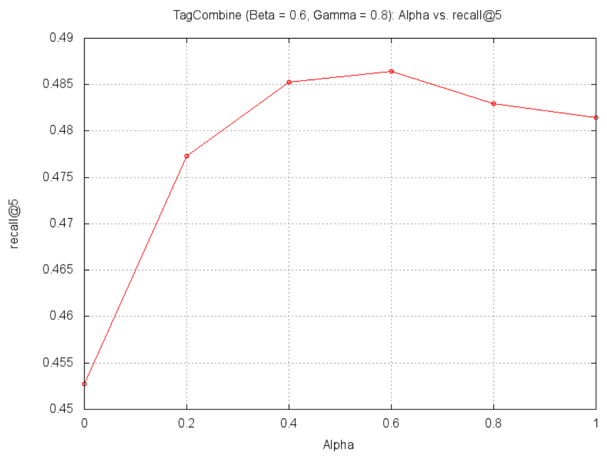


Fig. 6. Multilabel Classifier contribution to *TagCombine*.

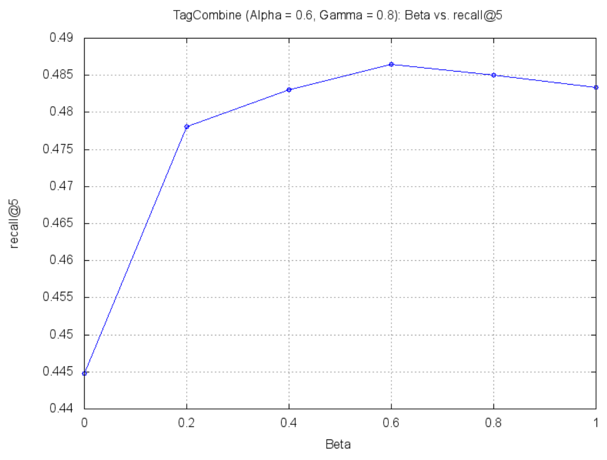


Fig. 7. Similarity Ranking contribution to *TagCombine*.

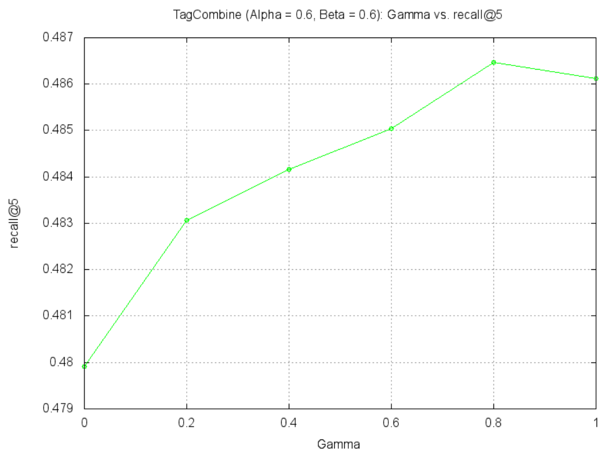


Fig. 8. Tag Affinity contribution to *TagCombine*.

One way we can measure the contribution of each individual component is to compare the $recall@5$ score when its corresponding parameters value is 0 (not used) and to the score when the parameter value is optimal. We can see that the Similarity Ranking component has the most significant contribution (≈ 0.041), then the Multilabel Classifier component (≈ 0.03), and finally the Tag Affinity component (≈ 0.006). So, when evaluating the improvements in our new tag recommendation model, in addition to comparing composite $recall@k$ scores, we can also look at the contribution of our added network component.

4.2 Network Based Improvements to TagCombine

In order to improve upon the tag recommendation accuracy of *TagCombine*, we sought to include components that utilized the structure of networks present in the StackOverflow data. Preliminary analysis of the graphs in Section 3 supported our original intuition that user and tag interactions on StackOverflow did exhibit behavior consistent with the interactions of users on more classical examples of social network sites. By leveraging this behavior, we constructed components that could accurately recommend tags using the structure present in the underlying networks of StackOverflow and without relying on textual data. Incorporating these improvements into *TagCombine* would allow us to add another dimension of information to the original algorithm and obtain results that leverage more of the data present in StackOverflow.

4.2.1 New Component: Our primary improvement to *TagCombine* was the addition of a fourth component that utilized the network structure stemming from the interaction of users with tags to predict the probability of each tag being associated with a given post. We begin by constructing the bipartite graph between users and tags described in Section 3.3.3. Given a post p, we compile a list of users who contributed to q and use the edge weights between the users and each tag to determine the tags ranking score. Define Q_i to be the edge weight between the user who posted the original question of p, and define the i^{th} tag and B_i to be the edge weight between the user who posted the best answer for p and the i^{th} tag. Next, for each comment in p, we sum up the edge weights between the user who posted the comment and the i-th tag, and define this value to be C_i. Now, for each answer in p, find the sum of the edge weights between the user who posted the answer and the i-th tag, and let this value be A_i. Using these values, the score for the i-th tag for post p, S_i is given by:

$$S_i = \kappa Q_i + \lambda B_i + \mu C_i + \nu A_i$$

where κ, λ, μ, and ν are constant weights evaluated by 10-fold cross validation to maximize tag recommendation accuracy. Calculating the score value S_i for each tag gives us the likelihood of each tag to be used to label p.

4.2.2 Modification to Similarity Component: The second improvement we made to TagCombine was to add features based on community detection to the similarity component. Using the graph in 3.3.1, we first run the SNAP implementation of BIGCLAM to group posts in StackOverflow into 1000 different communities, which we will label $C_1, \dots, C_{1000}$. For each community C_j, define the set of tag representatives R_j to be the set of tags that appear in at least 80% of the posts in C_j. In the similarity component of *TagCombine*, we now make the following modifications to the computation of the likelihood of assigning a tag g to a new question q.

$$L(g,q) = \frac{|\{i : g \in T_i\}| + \sum_i |\{j : p_i \in C_j,\ g \in R_j\}|}{\sum_i |T_i| + \sum_i |\{j : p_i \in C_j\}|}$$

That is, of the top 50 similar posts $p_1, \dots, p_{50}$, each community these posts belong in will also be able to contribute their tag representatives to the likelihoods of assigning tags to the new question q.

4.3 NetTagCombine Algorithm

In our proposed tag recommendation system, *NetTagCombine*, we add our new Network-Based Ranking Component, alongside the other components of *TagCombine*. Building upon the equation for *TagCombine* given in [1], for all tags t with respect to some post p, our new *NetTagCombine* score can be given by

$$\begin{aligned} NetTagCombine_p(t) = \alpha \times & MultiLabel_p(t) + \\ & \beta \times SimRank_p(t) + \\ & \gamma \times TagTerm_p(t) + \\ & \delta \times Network_p(t) \end{aligned}$$

where α,β,γ,δ $\in$ [0,1] represent the different weights of the components. Here, we have added the term of $\delta \times Network_p(t)$ to represent our new fourth component that uses the underlying network structure. To adjust for this fourth component, Algorithm 1 (next page) describes the pseudocode for *NetTagCombine*.

4.4 Results of NetTagCombine

After incorporating our new Network Based component and updating the Similarity Ranking component, we now had a working version of *NetTagCombine*, a tag recommendation system that used the underlying network structure of StackOverflow. Once again, we evaluated each individual component as standalone along with the entire *NetTagCombine* algorithm itself using 10-fold cross validation. This time, the addition of δ gave us an additional parameter to vary. In our runs of *NetTagCombine*, we found that $(\alpha, \beta, \gamma, \delta) = (0.4, 0.4, 1.0, 0.6)$ produced the best $recall@k$ scores. We give the results of our new or updated components as well as *NetTagCombine* itself. We have also included some previous numbers for easy comparison.

	$recall@5$	$recall@10$
New Similarity Ranking	0.462	0.550
Network	0.341	0.418
NetTagCombine	0.517	0.602
Our *TagCombine*	0.486	0.574
[1] Cited *TagCombine*	0.596	0.724

Just like our previous set of results, we also calculated the $recallAdj@k$ scores:

	$recallAdj@5$	$recallAdj@10$
New Sim. Ranking	0.594	0.709
Network	0.450	0.549
NetTagCombine	0.679	0.787
Our *TagCombine*	0.617	0.748
Cited *TagCombine*	0.596	0.724

Our new algorithm *NetTagCombine* shows significant improvements over our baseline numbers. For example, we once again measure the contribution of our Network Based component by holding α, β, and γ constant while varying our new parameter δ. Figure 9 shows the resulting plot.

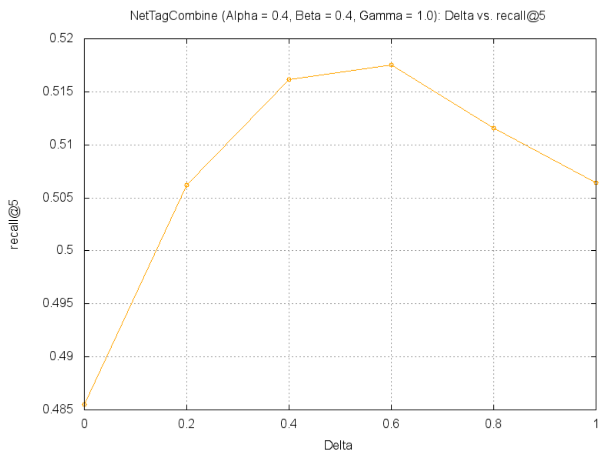


Fig. 9. Network contribution to *NetTagCombine*.

We can see that, for $recall@5$ scores, the Network Based component has a significant contribution ($\approx$ 0.031), which places it on par with the large contributions we found in our initial analysis of *TagCombine*. We also note how the influence of the underlying StackOverflow network through our updated Similarity Ranking component and Network Based component affects the differences between $recall@k$ and $recallAdj@k$ scores. For our implementation of *TagCombine*, there was about a 27% improvement from $recall@5$ to $recallAdj@5$ and about a 30% improvement from $recall@10$ to $recallAdj@10$.

Algorithm 1 *NetTagCombine* algorithm

```
1: $\alpha, \beta, \gamma, \delta \leftarrow 0$
2: for all posts $p$ do
3:     for all tags $t \in TAGS$ do
4:         Compute $MultiLabel_p(t)$, $SimRank_p(t)$, $TagTerm_p(t)$, and $Network_p(t)$
5:     end for
6: end for
7: for all $\alpha$ from 0 to 1, every time increment by 0.2 do
8:     for all $\beta$ from 0 to 1, every time increment by 0.2 do
9:         for all $\gamma$ from 0 to 1, every time increment by 0.2 do
10:            for all $\delta$ from 0 to 1, every time increment by 0.2 do
11:                Compute $NetTagCombine_p(t)$ for all tags $t$ on posts $p$
12:                Evaluate effectiveness of $(\alpha,\beta,\gamma,\delta)$ from $recall@k$ scores
13:            end for
14:        end for
15:    end for
16: end for
17: return Best $(\alpha,\beta,\gamma,\delta)$
```

Then, for *NetTagCombine*, we saw about a 31% improvement from $recall@5$ to $recallAdj@5$ and about a 30% improvement from $recall@10$ to $recallAdj@10$. Thus, while the addition of the Network Based component improves accuracy in all cases, it does particularly well when we predict the top 5 tags. This is even more promising, since it is more user-friendly to show fewer, but more accurate tag predictions.

To give some additional perspective on the performance of our algorithm, we calculated the $recall@k$ and $recallAdj@k$ scores for *NetTagCombine* after pruning the tagspace for tag synonyms as discussed in Section 3.2. This reduced the size of our tagspace by a small amount from 437 to 428. We give the previous results from using our "original" tagspace alongside our new results from using the "pruned" tagspace.

	Original	Pruned
$recall@5$	0.517	0.547
$recallAdj@5$	0.679	0.716
$recall@10$	0.602	0.654
$recallAdj@10$	0.787	0.851

As expected, the removal of tag synonyms from the tagspace allows *NetTagCombine* to perform better. Now, our algorithm does not suffer from making an "incorrect" prediction (such as `report` instead of `reporting`) when the root cause is a tag synonym group. An important point is that, while *NetTagCombine* can aide in the prevention of future tag synonyms from being introduced as discussed in Section 1, the existence of previous synonyms will hurt its performance and still must be pruned manually. If a synonym group already exists in the tagspace, then there is clearly no guarantee that *NetTagCombine* will always recommend one tag over another.

4.5 *Other Considerations*

We also tried to use BigClam to detect user communities on the graph in Section 3.3.2. For each user community U_i, we would compute the set of tag representatives S_i for U_i. A tag g would be in in S_i if at least 80% of the users in the community have posted a question or answered a question that was tagged with g. Then given a question posted by an existing user u, we would loop through all U_i the user belonged in and compute a likelihood for assigning a tag g to the user as

$$L(g,u) = \frac{|\{i : g \in S_i\}|}{\sum_i |S_i|}.$$

That is, the likelihood of assigning tag g to a question posted by user u would be proportional to the frequency the tag appeared as a representative of the communities u was in. However, adding this component did not improve the performance of *NetTagCombine*, as it is likely that any information produced by this network component is redundant with those outputted by our improvements in Sections 4.2.1 and 4.2.2.

5 Conclusion

In this paper, we developed a tag recommendation system *NetTagCombine* that combined the text-based machine learning tag recommendation techniques of *TagCombine* with our newly developed network-based tag recommendation techniques. By incorporating network structure and analysis into its tag recommendation system, *NetTagCombine* was able to yield significantly higher tag recommendation accuracy when compared to *TagCombine* on the StackOverflow dataset. Our results confirm our original intuition: StackOverflow and possibly other such user content based software information web sites have an underlying network structure that can be used to obtain additional information about posts and predict tags more accurately.

Our work demonstrates that network analysis can be employed to obtain data about the similarities present in content on information web sites without relying on information found within the body of a post. The network analysis we employed did not strictly focus on the textual content of posts, yet it was still able to produce accurate tag recommendations by analyzing the posts' positions in the StackOverflow networks. Since the information is obtained through entirely different inputs, there is little overlap between the data provided through network analysis and the data provided through text-based machine learning. Thus, we conclude that the use of network structure for tag recommendations is a promising avenue for the improvement of tag recommendation accuracy.

5.1 Future Work

As discussed in Section 2.3, we chose to evaluate our network-based improvements upon an implementation of *TagCombine* instead of the more recent *EnTagRec*. Future work, then, would be to apply our improvements to an implementation of *EnTagRec* and analyze whether *EnTagRec* improves in a similar manner to what we have seen in *TagCombine*. Furthermore, incorporating a network component into our model invites the possibility of a dynamic tag recommendation system. Tagging a post is not an action limited to post creation time. While this has already been addressed simply by the fact that we are using user interaction and collaboration – events that happen after post creation – to assist tag recommendations, we have so far only modeled the network based on one snapshot in time. Since there is public access to the entire log of post edit history, it would be interesting to see how our tag recommendations and $recall@k$ scores would change over time given snapshots in which more or less users had contributed to certain posts. Any discovered insights or trends would provide commentary on the effectiveness of future dynamic tag recommendations systems that constantly restructure their models along with the network itself.

References

[1] X. Xia, D. Lo, X. Wang, B. Zhou. Tag Recommendation in Software Information Sites. MSR, 2013.
[2] S. Wang, D. Lo, B. Vasilescu, A. Serebrenik. EnTagRec: An Enhanced Tag Recommendation System for Software Information Sites. ICSME, 2014.
[3] J. McAuley, J. Leskovec. Discovering Social Circles In Ego Networks. ACM TKDD, 2014.
[4] M. Girvan, M. E. J. Newman. Community Structure in Social and Biological Networks. National Academy of Sciences, 2002.
[5] A. Clauset, M. E. J. Newman, C. Moore. Finding Community Structure in Very Large Networks. APS, 2004.
[6] G. Palla, I. Derényi, I. Farkas, T. Vicsek. Uncovering the Overlapping Community Structure of Complex Networks in Nature and Society. Nature, 2005.
[7] J. Yang, J. Leskovec. Overlapping Community Detection at Scale: A Nonnegative Matrix Factorization Approach. WSDM, 2013.
[8] J. Yang, J. Leskovec. Community-Affiliation Graph Model for Overlapping Network Community Detection. ICDM, 2012.
[9] StackExchange Data Dump (September 26, 2014). Retrieved 2 November 2014. https://archive.org/details/stackexchange.

Appendix

Throughout the entire scope of this project, we worked on each component together in person rather than split up the work in portions to be done individually. For example, we always peer-programmed when writing code (to help decrease the number of bugs), and we collaborated on writing every paper using `writelatex.com`. We feel that it would be difficult to explicitly list out different individual contributions, and we believe that each member did an equal and fair amount of work. As such, we request that we each be given the same individual grade.