Finding Semantic Relationships in Folksonomies

1st Iman Saleh
Faculty of Computers and Information
Cairo University
Giza, Egypt
iman.saleh@fci-cu.edu.eg

2nd Neamat El-Tazi
Faculty of Computers and Information
Cairo University
Giza, Egypt
n.eltazi@fci-cu.edu.eg

***Abstract*—In this paper we study the problem of finding semantic relationships between folksonomy tags. We investigate different methods used to embed tags in the vector space and find similarities between them using word embedding vectors. We also present two new methods for embedding tags in the vector space utilizing labeled Latent Dirichlet Allocation (LDA) and Wikipedia category links. Related tags are grouped into communities using an overlapping community detection technique. In order to evaluate tag embedding methods, we use three different evaluation metrics, two of them do not require a ground truth dataset and the third is based on a manually created dataset of ground truth communities. Our results show that representing folksonomy tags using bag of words and embedding this representation in the vector space yields the best results compared to embedding co-occurring tags only or embedding tags along with textual content of tagged documents. We also compare between using word embedding, Latent Semantic Indexing (LSI), and LDA to find similarities between bag of words representations of tags. We show that word embedding outperforms LSI in one representation, while LDA is hard to beat.**

***Index Terms*—folksonomy, semantic similarity, word embedding, labeled LDA, Stack Overflow, Wikipedia**

I. Introduction

A folksonomy is a set of keywords that are assigned liberally to an information resource by the crowd. The process of tagging itself is referred to as collaborative tagging. Folksonomy tags assigned to an online resource contain information that complement information found in its body. They are more than just metadata, they are actually more content and highlight aspects of meaning that can be absent in the body of the resource. Folksonomies were designed to be informal without a specific hierarchy or relationships between tags. Since it is governed by the crowd, this means it is continuously changing in an unstructured way. As the number of tags increases, selecting an appropriate tag for a resource can be confusing due to the existence of synonymous, homonymous, and polysemous tags. Finding semantic relationships between tags helps in recommending appropriate tags to a user tagging a resource. Related tags can also be used to expand user queries and disambiguate tags when searching for resources that are assigned ambiguous tags. Moreover, they can give insights into the way a particular tag is used in a folksonomy.

This work is funded by Information Technology Academia Collaboration (ITAC) fund offered by Egyptian Information Technology Industry Development Agency (ITIDA). Grant No. PRP2015.R18.3.

One of the popular collaborative tagging platforms is Stack Overflow[1], a community Question Answering (cQA) forum for programming professionals. This forum is characterized by a large folksonomy of tags related to programming. It contains around 15,621,383 posts and 51,780 tags assigned by users[2]. One unique aspect about this platform is the continuous effort to maintain high quality posts and tags. Posts and answers are assigned scores by the crowd in order to encourage content editors to post high quality questions and answers. Another interesting aspect of Stack Overflow folksonomy is synonym control. Users can suggest synonymous tags and, after revision, suggestions can be accepted and synonymous tags get merged into a single tag. All these aspects ensure that the data contained in this platform is of high quality.

In this paper our focus is on finding semantic relationships between tags in Stack Overflow folksonomy. However, our methods can be applied to any other folksonomy where tags are applied to textual content. Similar tags can include synonyms or tags belonging to the same topic. For instance, "html" and "css" are both related to web design but they are not synonyms per se. We investigate different methods that can be used to capture semantic similarities between folksonomy tags. Methods investigated in this paper include finding relationships between tags based on the following representations (1) co-occurrence statistics of tags, (2) textual content of tagged resources and (3) tag signature, a piece of text that describes the meaning of a tag.

We utilize labeled LDA [1] and Wikipedia[3] category links in order to create tag signatures. Tags are embedded in the vector space based on each of these representations and then similarities between each pair of tags is calculated using word embedding vectors. Different representations used to embed tags in the vector space are explained in Section III-B. Communities of related tags are created afterwards using overlapping community detection techniques as explained in Section IV. We evaluate communities using different evaluation metrics and show that the third representation is the best to capture semantic similarity between folksonomy tags. We also compare the use of word embeddings, LSI [2] and LDA [3] to find similarities between bag of words representations

[1]https://stackoverflow.com/

[2]Statistics obtained from https://stackoverflow.com/questions and https://data.stackexchange.com/stackoverflow/query/new respectively

[3]https://en.wikipedia.org/wiki/Main_Page

of tags, where LSI and LDA are considered as baselines. Our results show that embedding outperforms LSI in one representation unlike LDA which outperforms word embedding in two different representations. Evaluation results and discussion are presented in Section V.

II. Related Work

An overview of a number of methods used to find related tags is presented in [4]. These methods include tag co-occurrence in documents, tag co-occurrence by the same user only, tag co-occurrence in the same document by the same user, and clustering documents based on their tags. Co-occurrence based methods assume that co-occurring tags are similar. Graph clustering techniques and co-occurrence statistics of tags are used in [5]–[9] to find semantically related ones. A different method utilizing co-occurrences to group related tags is introduced in [10]. They build a lattice of all tags such that co-occurrence relationships between them are captured. Next, they investigate two lattice partitioning techniques and how to rank partitions found. All these methods assume that relationships between tags are based on co-occurrences only and ignore the semantic aspects for grouping related tags.

Representing a tag using documents it tags, which is referred to as tag signature, and then using it to find related tags is studied in [11]–[14]. These studies use all the documents assigned a particular tag in order to represent a tag, which is computationally expensive when applied to tags of a cQA forum like Stack Overflow. For instance, the tag "multithreading" is assigned to 108,110 questions in Stack Overflow. This means that a tag is represented by hundreds of thousands of words. Therefore, a more efficient method needs to be employed to create tag signature.

External resources such as WordNet and Wikipedia are used in [15] to find related tags. Their study shows that associating tags with concepts helps in classifying resources. Since concepts are more general than terms, they reveal relationships between tags beyond term matching. Wikipedia contains more concepts than WordNet, and hence is more appropriate to find the meaning of a diverse set of tags. Tags are linked to Wikipedia category links or articles directly to help classify them. However, the number of Wikipedia category links is large and categories are diverse. Other methods to find semantic relationships between tags using Wikipedia category links need to be explored rather than linking tags to categories directly.

Word embedding techniques are used in [16] to find related tags. A number of word embedding techniques are applied to tags and the semantic quality of generated embeddings is evaluated. To the best of our knowledge, this is the only study that utilizes embedding vectors to group related tags. However, it is based on creating embeddings of tags based on their co-occurrences only. In our work we investigate other methods used to embed folksonomy tags in the vector space. In addition to utilizing co-occurrence statistics, we utilize textual content of tagged resources and tag signatures to embed tags in the vector space. The first two methods assume that a tag is a single word that is embedded in the vector space. When tags are embedded using their co-occurrences, this means that tags only are embedded in a vector space where related tags are placed in close proximity to each other. Using textual content of tagged resources means that a tag is embedded in a space containing both tags and words in a tagged resource. On the other hand, tag signatures assume that each tag is a self contained document consisting of a bag of words.

We propose two efficient methods to create tag signatures that were not studied in literature for this task previously. Both methods use keywords produced by labeled LDA for each tag and keywords of Wikipedia category links related to a tag to create signatures. After embedding tags in the vector space using explained methods, an overlapping community detection technique is used to create communities of related tags. The quality of generated communities is evaluated using different evaluation metrics.

Results show that representing a tag using a signature is better than other methods utilizing co-occurrence statistics or textual content of tagged resources. We also compare between different techniques used to embed tag signatures in the vector space, namely word embedding and LSI, to find similarities between them. We show that word embedding outperforms LSI when a tag is represented using labeled LDA keywords. Moreover, we compare the use of word embedding and LDA topic probability distributions to find similarities between tag signatures and show that LDA is hard to beat. This study is the first study that compares between different methods of embedding tags in the vector space and finding similarities between them.

III. Representing Folksonomy Tags and Finding Similar Ones

In this section we briefly explain the word embedding model used in our experiments to embed tags and words in the vector space. Then we explain different methods utilized to embed a tag in the vector space. Finally, we show how communities of related tags are created and the similarity metric utilized to group tags into communities.

A. Word Embedding Model

Word embedding models belong to predictive distributional semantic models that try to predict a word from its neighbour words. Their goal is to embed words in the vector space in order to encode semantic relationships between them. Moreover, more complex semantic relationships between words can be derived by applying algebraic operations to word vectors. For example, $Vector(\text{“}Man\text{”}) - Vector(\text{“}Woman\text{”}) + Vector(\text{“}King\text{”})$ results in a vector that is close to $Vector(\text{“}Queen\text{”})$ [17]. Theses models have been shown to boost the performance of several tasks. Skip-gram and continuous bag-of-words (CBOW) models introduced by [18] are based on the use of a neural network structure to learn word representations. Skip-gram models predict the context of a word given the word itself, while

CBOW models predict a word given its context. These methods however have the disadvantage of not operating directly on the co-occurrence statistics of a corpus. They scan a context window across the whole corpus, and fail to take advantage of the repetitions present in the data [19]. Another word embedding model is **Gl**obal **Ve**ctors of word representations (Glove) [19], which captures the global statistics of a corpus to represent words as vectors. Glove is a specific weighted least squares model that uses global words co-occurrence counts in training, and thus makes efficient use of corpus statistics. It outperforms other models in several word similarity tasks [19]. Since finding similarity between words representing tags is our main goal in this paper, we use Glove to obtain word vector representations in our experiments.

B. Embedding Tags in the Vector Space

In this subsection, we explain different methods used to embed tags in the vector space in our study. We first explain three baseline models and then introduce two new models which were not studied in literature before for this task.

1) Co-occurrence Statistics (E_{cotags}): We create Glove word vectors for each tag using a corpus of co-occurring tags in Stack Overflow posts. Since co-occurrence relationship can indicate semantic similarity, we expect this representation to produce reasonable results in finding related tags. In this representation only tags are embedded in the vector space.

2) Textual Content of Tagged Resources (E_{tag}): In this representation a tag is considered one of the content words of a document. Tags not only are metadata but also complementary data to document text. Glove word vectors are trained using both text and tags of Stack Overflow posts concatenated. Embedding vectors of tags are used subsequently to find similarities between them.

3) Tag Excerpts (E_{ex}): A tag excerpt is a short piece of text that describes the meaning of a tag concisely. It is created by the collaborative tagging platform and therefore it is an accurate description of the meaning of a tag. An example of a tag excerpt is shown in Figure 1. Tag excerpts can be used to identify related tags. In this representation, Glove word vectors of stemmed content words of an excerpt are used to find similarities between tags. This representation assumes that a tag is a document consisting of a bag of words.

youtube
YouTube is a video-sharing website on which users can upload, share, and view videos.

Fig. 1: Excerpt of "youtube" tag provided by Stack Overflow website

4) Keywords of Labeled LDA (E_{llda}): Labeled LDA is a probabilistic graphical model that is considered one variation of LDA topic modeling technique. It constrains LDA by defining a one to one correspondence between tags and topics [1]. This means that the model learns correspondences between tags and words in a corpus directly. A document is modeled as a mixture of underlying topics and each word is generated from a topic. The number of topics is equivalent to the number of unique tags in a corpus. Labeled LDA provides word-topic assignments that can be used to represent tags, where a topic represents a tag. Top N keywords assigned to each tag, K_N^T, are used in our experiments as a tag signature. These keywords effectively summarize documents where a tag is present. Word embedding vectors of keywords are used to find similarities between tags in the vector space.

5) Keywords of Wikipedia Category Links (E_{wiki}): We use Wikipedia as an external resource since it contains lots of articles about programming technologies that overlap with Stack Overflow tags. Wikipedia category links related to each tag are collected from wiki descriptions provided by Stack Overflow[4]. Wiki descriptions are first searched for any Wikipedia link. These links are considered as Wikipedia pages related to Stack Overflow tags. For instance, Wikipedia articles extracted from wiki descriptions of tags "c++" and "python" are shown in Table I. We refer to these articles as $WIKI_{sof}$. Next, category links of each extracted article are retrieved using DBPedia SPARQL endpoint[5]. We refer to these category links as CAT_{sof}.

Category links of Wikipedia articles can be considered as "topics" to which these articles belong. Therefore, we think they can be useful in identifying related Stack Overflow tags, since each tag is related to a number of articles. Moreover, we analyzed both $WIKI_{sof}$ and CAT_{sof}, and found that the number of category links shared among Stack Overflow tags is generally larger than the number of shared Wikipedia articles. For instance, there are 151 tags that share "Cross-platform_software" category link. On the other hand, the maximum number of tags that share a Wikipedia article, "Firefox", is 15. For that reason we prefer to use category links in our experiments rather than articles. Category links related to tags "c++" and "python" are shown in Table I.

Category links irrelevant to programming technologies, such as "Don't_repeat_yourself" and "Dutch_computer_scientists", are manually removed from CAT_{sof} to avoid representing Stack Overflow tags with words irrelevant to their main topic. For each Wikipedia category link, a set of N keywords are produced using Labeled LDA. Glove word vectors of keywords of category links related to a Stack Overflow tag are used to find similarity relationships with other tags in the vector space.

C. Calculating Similarity Scores between Pairs of Tags

The distance between a pair of tags using E_{cotags} and E_{tag} methods is the Euclidean distance between their embedding vectors, since both methods do not have a bag of word representation for a tag. For the remaining methods, we utilize Word Mover's Distance (WMD) metric [20] in order to calculate the distance between pairs of tag signatures. WMD utilizes word embedding vectors such that the semantic similarity between individual word pairs in two documents is

[4]Downloaded from https://data.stackexchange.com/stackoverflow/query/new
[5]http://wiki.dbpedia.org/

Stack Overflow tag	Wikipedia article	Category links
c++	"C++"	"Programming_language_topics"
	"Type_safety"	"Class-based_programming_languages"
	"Type_system#Static_typing"	"C++_programming_language_family"
python	"Don't_repeat_yourself"	"Class-based_programming_languages"
	"Python_(programming_language)"	"Object-oriented_programming_languages"
	"Dynamic_programming_language"	"Dutch_computer_scientists"
	"Guido_van_Rossum"	

TABLE I: Wikipedia articles and category links related to "c++" and "python"

used to calculate distance between them. This method has several advantages (1) it is hyper-parameter free (2) it is interpretable and straightforward, the distance between two documents is broken down into distances between their words (3) it incorporates the valuable information in word embedding models. WMD allows each $word_i$ in a tag signature t to be transformed to any $word_j$ in another tag signature t'. The distance between $word_i$ and $word_j$ is measured using Euclidean distance as follows $c(i,j) = ||x_i - x_j||_2$, where x_i and x_j are embedding vectors of words i and j. In other words, the distance between two tag signatures is the minimum cumulative distance that words from one signature need to travel to match the document cloud of another signature. It is worth mentioning that WMD metric can calculate the distance between a pair of tag signatures even if their lengths are not equivalent.

D. Grouping Related Tags

After embedding each tag in the vector space and using WMD to measure similarity between pairs of tags, we need to create communities of related tags. Tags can be represented as a set of connected graph nodes, where edge weights represent similarity between each couple of nodes. Community detection techniques are an ideal method to find communities of related tags. Using overlapping community detection techniques is more realistic since Stack Overflow folksonomy contains tags that can belong to multiple communities. For instance, "windows-phone-8" is related to speech and audio community and also mobile technologies community. We explored different implementations of overlapping community detection techniques available online. For our experiments we needed a technique that is (1) fast enough[6] to process a large number of graph edges in a reasonable time, and (2) uses weights of graph edges to find communities, since semantic similarity between tags is based on weights. Order Statistics Local Optimization Method[7] (OSLOM) [21] was the most suitable for our task. Therefore, we report the results of our experiments using OSLOM tool.

IV. Datasets and Experiments

The statistics of datasets used are summarized in Table II. All datasets are tokenized and stemmed using Stanford CoreNLP toolkit [22]. We refrain from stemming tags in our experiments because we found that it can merge concepts that differ in meaning together. For instance, "io" and "ios" tags will both become "io" after stemming although each refers to a different concept: input/output and iPhone OS. Stack Overflow posts and Wikipedia articles containing less than five words are discarded.

[6]A tool that is fast enough would take at most 24 hours to produce tangible results using an Intel Core i7 processor and 15 GB RAM laptop.

[7]http://www.oslom.org/

A. Training Labeled LDA

We train MALLET implementation of Labeled LDA [23] using two datasets: a sample of Stack Overflow posts and a sample of Wikipedia articles. In order to obtain a dataset that can be used to train Labeled LDA in a reasonable time, we only use posts that contain tags linked to at least one Wikipedia category link and have an excerpt. Keywords of such tags are of interest to be able to compare different tag representation methods. We filtered these posts further using their community assigned scores such that posts assigned a score greater than or equal to zero are included. This step ensures that posts used are of a reasonable quality. A post in our experiments consist of title, question, in addition to best answer if available. The total number of posts used after filtering is 2,935,329 posts. We refer to this dataset as SOF_{llda}.

The second dataset used to train Labeled LDA consists of Wikipedia abstracts of articles and their category links as labels. Abstracts are filtered such that only those assigned at least one category link in CAT_{sof} are used. This way we ensure that we obtain keywords representing category links related to Stack Overflow tags. The total number of abstracts used is 115,430. All abstracts are obtained using DBPedia SPARQL endpoint. We refer to the latter dataset as $WIKI_{llda}$. The top $N = 20$ keywords of Labeled LDA output are used to represent a tag or a category link. The number of top words that was shown to capture important vocabulary of a topic is 10-15 as shown in [1], [3]. We use 20 words to ensure that all relevant words are used.

B. Training Word Embeddings

After training Labeled LDA, each Stack Overflow tag is represented using K_{20}^T keywords. In order to be able to find similarities between tags, we represent each keyword using Glove word vectors. Since training word embeddings is better if a large dataset is used, we train Glove using all Stack Overflow posts that have a community assigned score greater than or equal to zero. Title of post, tags, question as well as the best answer, if available, are combined together in training

	Num. of posts/articles	Vocabulary size	Num. of tags/category links
SOF_{llda}	2M+	111K+	3K+
$WIKI_{llda}$	115K+	149K+	24K+
SOF_{we}	10M+	212K+	46K+
$WIKI_{we}$	4M+	357K+	1M+

TABLE II: Datasets used to train Labeled LDA and word embeddings

data. The total number of posts used is 10,854,607. We refer to this dataset as SOF_{we}.

Word vectors of Wikipedia category links are also needed in order to use them to represent each Stack Overflow tag. We think that this representation is appropriate since Wikipedia category links are numerous and diverse, 3,657 Wikipedia category links are linked to 3,616 Stack Overflow tags. Therefore, representing category links using word vectors will help place category links close to each other in meaning in close proximity in the vector space. We train Glove using Wikipedia dataset that consists of 4,841,864 long abstracts. We refer to this dataset as $WIKI_{we}$. Category links were not included in the training because their number was large, more than one million, and including them would affect training speed. We also do not need their vectors in our experiments.

Word and tag vectors for E_{ex} and E_{tag} are obtained from the output of training Glove using SOF_{we} dataset. In this dataset, tags are part of the content words and each tag has a word vector representation. Vectors used in E_{cotags} are created using co-occurring tags in SOF_{we} dataset. For this representation we train Glove using co-occurring tags only to create tag embeddings. We set the vector size in word embedding training to $M = 800$, and the window size is set to 10. These values are shown to perform best in classifying short documents [24].

C. Finding Related Tags

The dataset we use consists of 2,154 tags. In order to be able to compare different methods, these tags must have the following characteristics (1) each tag has an excerpt in Stack Overflow wiki, (2) each tag must be represented by exactly 20 keywords produced by Labeled LDA model to make sure its meaning is captured properly, (3) each tag must be linked to at least one Wikipedia category link that is represented by 20 keywords produced by Labeled LDA model. The similarity between 206,288 pairs of co-occurring tags is calculated using WMD similarity metric. We use gensim[8] implementation of WMD similarity [20], [25], [26]. Since we need to find clusters of related tags, similarity scores between tags are needed rather than distance, therefore we use the negative of WMD score. The negative is $\frac{1}{(1+WMD)}$, as implemented in gensim.

Next, overlapping communities are extracted using OSLOM tool. The technique implemented by the tool has a number of parameters: (1) t: the p-value used for optimizing statistical significance for clusters, (2) cp: coverage parameter that is used to decide between taking some clusters or their union and (3) r: the number of runs used to find clusters. First, we run the community detection technique using different values for each parameter. t is assigned to values $\{0.1, 0.3, 0.5, 0.9\}$. cp is assigned to values $\{0.2, 0.5, 0.8\}$. r is assigned to values $\{10, 20, 30, 40, 50\}$. We test our community detection technique with all combinations of these values. To determine which values are the best, we investigate modularity, clustering coefficient and F1 measure, explained in Section V. In most cases the best values where achieved when $t = 0.5$ and $cp = 0.8$, therefore, we used these values in our experiments. We report evaluation results using $r = \{10, 20, 30, 40, 50\}$. In order to validate evaluation scores and calculate statistical significance of improvement where appropriate, communities of each run in our experiments are generated five times using different seeds for the community detection algorithm. Our evaluation results are the average scores of the five runs.

[8]https://radimrehurek.com/gensim/similarities/docsim.html

V. Evaluation and Discussion

Since a complete gold set of Stack Overflow communities is not publicly available, we first use modularity [27] and clustering coefficient [28] metrics to evaluate the quality of generated communities. However, for the sake of completeness, we also create a manual subset of communities and evaluate them using F-measure. Our scores are the average scores of the five runs generated by the community detection algorithm for each value of r for different methods. One tailed paired t-test was used to measure statistical significance of improvement at 95% level of confidence.

Modularity computes the difference between the number of edges in a community and the number of edges expected to appear in the same sub-graph in the *"null-model"*, a model used to verify if a graph has a community structure. The higher the modularity, the greater the difference between edge density in a sub-graph and expected value in the null-model. We calculate modularity Q for a community structure R for our weighted graph using the following equation [29]:

$$Q(R) = \sum_{i=1}^{cn} \left[\frac{I_i}{m} - \left(\frac{2I_i + O_i}{2m} \right)^2 \right] \quad (1)$$

where I_i is the sum of the weights of edges between nodes in a community C_i, m is the sum of the weights of edges in the graph, O_i is the sum of weights on edges from nodes in C_i to nodes outside, and cn is the total number of communities.

Clustering coefficient measures local cohesiveness of a community and is defined for every node $v \in C_i$ as the fraction of its connected neighbours. We use the weighted clustering coefficient introduced in [30] and calculate the average score for a community structure as follows:

$$cc(R) = \frac{1}{cn}\sum_{i=1}^{cn}\left(\frac{1}{|C_i|}\sum_{v\in C_i}\frac{1}{s_i(k_i-1)} \sum_{j,h}\frac{(w_{ij}+w_{ih})}{2}a_{ij}a_{ih}a_{jh}\right) \quad (2)$$

where s_i is the strength of node v, which is the total weight of its connections and k_i is the degree of node v. a_{ij}, a_{ih}, a_{jh} is a triangle in C_i.

Modularity and clustering coefficient do not require a gold dataset. However, F-measure is more commonly used and its results are easier to understand. Therefore, we create 7 communities manually, containing 431 tags, by defining topics then finding tags belonging to each topic to calculate F1 scores of generated communities. Topics included are "database", "algorithms", "machine learning", "memory management", "mobile development", "speech and video" and "web development". F1 scores are calculated for pairs of tags as explained in [31] as follows.

- **True positive (TP):** a pair is present in at least one gold community and one community produced by our method.
- **False positive (FP):** a pair is not present in any gold community but is present in at least one output community.
- **True negative (TN):** a pair is not present in any gold community or output community.
- **False negative (FN):** a pair is present in at least one gold community and is not present in any output community.

A. Word Embedding Results

Table III shows scores of both evaluation metrics. The highest modularity scores for most methods are achieved when $r = 10$. Increasing the number of runs beyond this value does not result in improvement except for E_{ex} and E_{llda} which reach their best modularity scores at $r = \{20, 30\}$ respectively. The same applies to clustering coefficient. Increasing the number of runs also results in higher scores for E_{ex} and E_{llda} methods at $r = \{50, 30\}$ respectively.

Results of modularity show that E_{wiki} is the best method since no other method outperformed it using different values of r. On the other hand, E_{wiki} method outperforms E_{tag}, E_{ex} and E_{llda} at $r = \{20, 10, 10\}$. Although E_{tag} and E_{llda} achieved the highest scores at different values of r, both were outperformed by other methods. For instance, E_{ex} and E_{wiki} outperform E_{tag} at $r = 20$. E_{tag} outperforms E_{llda} at $r = \{10, 50\}$ while E_{llda} outperforms it at $r = \{20, 30\}$. Clustering coefficient results also agree with modularity. All methods failed to outperform E_{wiki}, while the latter outperformed E_{tag}, E_{ex} and E_{llda} at $r = \{20, 10, 10\}$. Although E_{cotags} has the highest clustering coefficient scores at $r = 10$, E_{llda} outperformed it at $r = 40$. E_{llda} also has the highest scores at $r = \{30, 40\}$, however, E_{cotags}, E_{tag} and E_{wiki} outperform it at $r = 10$.

Table IV shows F1 scores for $r = \{10, 20, 30, 40, 50\}$. Increasing the number of runs also results in improvement for E_{ex}, E_{llda} and a slight improvement in E_{wiki}. This evaluation metric shows that E_{ex} and E_{llda} are the best methods. E_{ex} outperforms E_{cotags} and E_{tag} at $r = \{30, 40\}$. E_{llda} outperforms E_{cotags} and E_{tag} at $r = \{30, 40\}$. No other method outperformed E_{ex}. On the other hand, E_{llda} was outperformed by E_{tag} when $r = 10$. E_{wiki} did not outperform any other method and no other method outperformed it as well.

Overall, the three evaluation metrics introduced differ in the way each use to asses the quality of communities produced by each method. However, we generally find that the best methods for finding related tags according to them are E_{ex}, E_{llda} and E_{wiki}. Therefore, we conclude that using bag of words to represent folksonomy tags is better than using co-occurrence statistics or representing them as part of the textual content of documents they tag.

B. Comparing Word Embedding to LSI and LDA

We compare the use of LSI and word embedding to embed tag signatures in the vector space and find similarities between them. We use gensim implementation [32] with default parameters to train LSI using two corpora: SOF_{we} and $WIKI_{we}$. Tag signatures are mapped using trained model into LSI vector space. Euclidean similarity, calculated as the inverse of euclidean distance, is used to measure similarity between LSI vectors of tags. Table V shows F1 scores of LSI compared to word embedding. Using word embedding to embed tag signatures created using labeled LDA (E_{llda}) outperforms embedding those signatures using LSI (LSI_{llda}) when $r = 30$.

We also compare the use of word embedding and LDA topic probability distributions in order to find similarities between tag signatures. We train LDA using SOF_{llda} and $WIKI_{llda}$ corpora with 100 topics and default parameters. Next, we obtain probability distributions for each tag signature. We find similarity between probability distributions using Hellinger similarity, the inverse of Hellinger distance [33]. Table V shows F1 scores of LDA compared to word embedding. Using LDA with labeled LDA and Wikipedia tag signatures, LDA_{llda} and LDA_{wiki}, outperforms word embedding results, E_{llda} and E_{wiki}, when $r = 10$. LDA is more focused on modeling topics in documents compared to word embeddings, which embed single words in the vector space and then find distances between documents based on word distances. We expect that this is the reason why LDA outperformed word embeddings.

C. Error Analysis

In order to analyze results further, we calculated confusion matrix for the best word embedding model, E_{llda} at $r = 30$, shown in Table VI. In this matrix, rows correspond to gold communities and columns correspond to predicted communities. We notice that the number of false negative pairs is large, 8,287 pairs, compared to 3,359 false positive pairs. We further analyze communities of false negative and false positive pairs as illustrated in Table VII. This table shows confusion between gold and output communities if it occurred > 100 times. Left table shows gold communities, their sizes,

Q(R)	$r=10$	$r=20$	$r=30$	$r=40$	$r=50$
E_{cotags}	0.148	0.138	0.142	0.138	0.143
E_{tag}	**0.147**	0.136	0.140	0.138	**0.146**
E_{ex}	0.141	**0.142**	0.142	0.139	0.143
E_{llda}	0.142	**0.147**	**0.147**	0.141	0.143
E_{wiki}	**0.148**	**0.144**	0.145	0.139	0.143

cc(R)	$r=10$	$r=20$	$r=30$	$r=40$	$r=50$
E_{cotags}	**0.040**	**0.036**	0.037	0.034	0.038
E_{tag}	**0.040**	0.034	0.034	0.036	0.038
E_{ex}	0.034	0.036	**0.037**	0.036	0.038
E_{llda}	0.035	0.037	**0.039**	**0.038**	0.036
E_{wiki}	**0.038**	**0.036**	0.037	0.035	0.037

TABLE III: Modularity and clustering coefficient scores. Bold values are statistically significant higher than at least one other method in the same run.

F1	$r=10$	$r=20$	$r=30$	$r=40$	$r=50$
E_{cotags}	54.53	50.49	48.94	50.01	54.22
E_{tag}	**56.15**	49.48	50.85	50.36	53.42
E_{ex}	54.96	50.60	**54.81**	**54.40**	52.62
E_{llda}	49.76	51.10	**56.78**	**53.47**	52.54
E_{wiki}	52.46	51.30	52.13	51.30	53.09

TABLE IV: F1 scores of communities for $r = \{10, 20, 30, 40, 50\}$. Bold values are statistically significant higher than at least one other method in the same run.

	$r=10$	$r=20$	$r=30$	$r=40$	$r=50$
E_{ex}	54.96	50.60	54.81	54.40	52.62
E_{llda}	49.76	51.10	<u>56.78</u>	53.47	52.54
E_{wiki}	52.46	51.30	52.13	51.30	53.09
LSI_{ex}	49.21	54.32	51.98	50.03	52.76
LSI_{llda}	55.84	51.98	52.75	53.88	50.95
LSI_{wiki}	53.07	54.59	53.07	49.30	52.93
LDA_{ex}	53.82	49.83	54.47	53.30	52.78
LDA_{llda}	**55.69**	53.76	52.24	51.04	50.88
LDA_{wiki}	**56.42**	51.52	50.59	50.24	51.12

TABLE V: F1 scores of communities, bold values indicate results that are statistically significant better comparing embeddings and LDA. Underlined values indicate results statistically significant better comparing embeddings and LSI.

and the number of tag pairs that were not successfully assigned to each community, i.e. false negatives. Right table shows communities produced by our method, their sizes, and the number of tag pairs erroneously assigned to one cluster, i.e. false positives.

We notice that the community containing web development related tags has the largest number of erroneous pairs. This is due to the large number of tags in this community and the diversity of tags related to web development. Also, community detection algorithm can capture relationships between tags that may not be captured using manual annotation. Tables VIII and IX illustrate confusion between different communities for false positives and false negatives. To produce both tables we manually assigned a title to communities produced by our method. In Table VIII, the first column lists gold communities of false positive pairs of tags. The second column lists communities produced by our method. Third column lists number of times communities in first column were confused for communities in second column. In most cases, confused communities are actually related to each other, for instance, "Mobile" community and "Speech and Audio" communities are closely related and are confused for "Mobile" community. Also, "Mobile" and "Web Development" are confused for "Web Development", both are closely related.

	Same community	Different communities
Same community	9,548	8,287
Different communities	3,359	59,006

TABLE VI: Confusion matrix of community detection results.

The same applies to confusion in false negative pairs shown in Table IX. We even notice that our method produced two different "Database" communities, and this resulted in 360 pairs of false negatives although pairs of tags were correctly assigned to database communities. These results show that it is difficult to produce accurate gold communities manually for our task. Most Stack Overflow tags are also related to each other which makes the task of producing gold communities even more difficult.

VI. Conclusion and Future Work

In this paper, we study the problem of finding semantic similarities between folksonomy tags. We investigate different methods used to embed tags in the vector space in order to create communities of semantically related tags. To the best of our knowledge, this is the first study investigating different tag embedding methods. Methods studied in this paper utilize co-occurrence statistics of tags, textual content of tagged documents, in addition to creating tag signatures for each tag in order to embed a tag in the vector space. We present two new methods for creating tag signatures utilizing labeled LDA and Wikipedia category links. Studied methods are evaluated using three different evaluation metrics. Our results show that embedding tags in the vector space using signatures is the best method compared to other studied methods. We also show that word embedding of signatures outperforms LSI in finding related tags when labeled LDA is used to create tag signatures, while LDA outperforms word embedding when used with labeled LDA and Wikipedia tag signatures.

For the future, we aim at investigating word embedding methods focused on documents rather than words to embed tag signatures in the vector space.

References

[1] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning, "Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2009, pp. 248–256.

[2] G. W. Furnas, S. Deerwester, S. T. Dumais, T. K. Landauer, R. A. Harshman, L. A. Streeter, and K. E. Lochbaum, "Information retrieval using a singular value decomposition model of latent semantic structure," in *Proceedings of the 11th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: ACM, 1988, pp. 465–480.

Community	Community Size	Num. of FN Pairs
Web Development	157	4,950
Database	66	981
Algorithms	64	795
Mobile	44	536
Memory Management	42	531
speech and audio	46	477

Community	Community Size	Num. of FP Pairs
Web Development	346	969
Mobile	132	811
Machine Learning	322	539
Programming	247	335
OS	255	233
Design for Mobile App.	174	183
Programming Concepts	198	154

TABLE VII: Statistics of FN pairs (left) and FP pairs (right) produced by E_{llda}, $r = 30$

Actual Communities	Produced Community	Num. of Pairs
Mobile, Speech and Audio	Mobile	593
Web Development, Mobile	Web Development	405
Machine Learning, Algorithms	Machine Learning	368
Web Development, Speech and Audio	Web Development	304
Web Development, Database	Programming	252
Web Development, Memory Management	Web Development	102
Web Development, Algorithms	Web Development	102

TABLE VIII: Confusion between communities for FP pairs

Actual Communities	Produced Community	Num. of Pairs
Web Development	Programming, Web Development	2,500
Web Development	Web Development, Editors	400
Database	Database 1, Database 2	360
Algorithms	Programming Basic Concepts, Machine Learning	329
Web Development	Mobile Design, Web Development	296
Database	Programming, Database	280
Memory Management	Programming Basic Concepts, OS	242
Speech and Audio	Mobile, OS	180
Algorithms	Security, Machine Learning	141
Algorithms	Mobile Design, Machine Learning	141

TABLE IX: Confusion between communities for FN pairs

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[4] C.-m. Au Yeung, N. Gibbins, and N. Shadbolt, "Contextualising tags in collaborative tagging systems," in *Proceedings of the 20th ACM Conference on Hypertext and Hypermedia*. New York, NY, USA: ACM, 2009, pp. 251–260.

[5] G. Begelman, P. Keller, F. Smadja *et al.*, "Automated tag clustering: Improving search and exploration in the tag space," in *collaborative web tagging workshop in conjunction with WWW2006, Edinburgh, Scotland*, 2006, pp. 15–33.

[6] P. Heymann and H. Garcia-Molina, "Collaborative creation of communal hierarchical taxonomies in social tagging systems," Stanford, Tech. Rep., 2006.

[7] L. Specia and E. Motta, "Integrating folksonomies with the semantic web," in *Proceedings of the 4th European Conference on The Semantic Web: Research and Applications*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 624–639.

[8] R. Li, S. Bao, Y. Yu, B. Fei, and Z. Su, "Towards effective browsing of large scale social annotations," in *Proceedings of the 16th International Conference on World Wide Web*. New York, NY, USA: ACM, 2007, pp. 943–952.

[9] J. Radelaar, A.-J. Boor, D. Vandic, J.-W. Van Dam, and F. Fasincar, "Improving search and exploration in tag spaces using automated tag clustering," *Journal of Web Engineering*, vol. 13, no. 3-4, pp. 277–301, 2014.

[10] M. Eftekhar and N. Koudas, "Partitioning and ranking tagged data sources," in *Proceedings of the 39th international conference on Very Large Data Bases*. VLDB Endowment, 2013, pp. 229–240.

[11] G. Solskinnsbakk and J. A. Gulla, "Mining tag similarity in folksonomies," in *Proceedings of the 3rd International Workshop on Search and Mining User-generated Contents*. New York, NY, USA: ACM, 2011, pp. 53–60.

[12] S. Wang, D. Lo, and L. Jiang, "Inferring semantically related software terms and their taxonomy by leveraging collaborative tagging," in *International Conference on Software Maintenance*. IEEE Computer Society, 2012, pp. 604–607.

[13] A. P. Garca-Plaza, A. Zubiaga, V. Fresno, and R. Martnez, "Reorganizing clouds: A study on tag clustering and evaluation," *Expert Systems with Applications*, vol. 39, no. 10, pp. 9483–9493, 2012.

[14] J. Zhu, B. Shen, X. Cai, and H. Wang, "Building a large-scale software programming taxonomy from stackoverflow," in *The 27th International Conference on Software Engineering and Knowledge Engineering*. Research Inc. and Knowledge Systems Institute Graduate School, 2015, pp. 391–396.

[15] A. Tommasel and D. Godoy, "Semantic grounding of social annotations for enhancing resource classification in folksonomies," *Journal of Intelligent Information Systems*, vol. 44, no. 3, pp. 415–446, 2015.

[16] T. Niebler, L. Hahn, and A. Hotho, "Learning word embeddings from tagging data: A methodological comparison," in *Proceedings of the Lernen. Wissen. Daten. Analysen workshop*, 2017.

[17] T. Mikolov, S. W.-t. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2013.

[18] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *International Conference on Learning Representations Workshop*, 2013.

[19] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014.

[20] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger, "From word embeddings to document distances," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning*. JMLR.org, 2015, pp. 957–966.

[21] A. Lancichinetti, F. Radicchi, J. J. Ramasco, and S. Fortunato, "Finding statistically significant communities in networks," *PLOS ONE*, vol. 6, pp. 1–18, 2011.

[22] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, "The stanford corenlp natural language processing toolkit." in *Annual Meeting of the Association for Computational Linguistics (System Demonstrations)*, 2014, pp. 55–60.

[23] A. K. McCallum, "Mallet: A machine learning for language toolkit," 2002, http://mallet.cs.umass.edu.

[24] X. Yang, C. MacDonald, and I. Ounis, "Using word embeddings in twitter election classification," *Information Retrieval Journal*, pp. 1–25, 2017.

[25] O. Pele and M. Werman, "A linear time histogram metric for improved sift matching," in *Proceedings of the 10th European Conference on Computer Vision: Part III*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 495–508.

[26] ——, "Fast and robust earth mover's distances," in *IEEE 12th International Conference on Computer Vision*, 2009, pp. 460–467.

[27] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, p. 026113, 2004.

[28] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440–442, 1998.

[29] M. E. Newman, "Analysis of weighted networks," *Physical review E*, vol. 70, no. 5, p. 056131, 2004.

[30] A. Barrat, M. Barthélemy, R. Pastor-Satorras, and A. Vespignani, "The architecture of complex weighted networks," *Proceedings of the National Academy of Sciences*, vol. 101, no. 11, pp. 3747–3752, 2004.

[31] A. Banerjee, C. Krumpelman, J. Ghosh, S. Basu, and R. J. Mooney, "Model-based overlapping clustering," in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. New York, NY, USA: ACM, 2005, pp. 532–537.

[32] R. Řehůřek, "Subspace tracking for latent semantic analysis," in *ECIR 2011: Advances in Information Retrieval*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 289–300.

[33] M. S. Nikulin, "Hellinger distance," *Encyclopedia of mathematics*, vol. 151, 2001.