Dynamic Tag Recommendation in High Dimensional Systems

Behrens, Hans.
Department of Computer Science
Ira A. Fulton School of Engineering
Arizona State University
Tempe, Az - 85281
hwbehren@asu.edu

Garg, Yash.
Department of Computer Science
Ira A. Fulton School of Engineering
Arizona State University
Tempe, Az - 85281
ygarg@asu.edu

Kadambi, Pradyumna.
Department of Computer Science
Ira A. Fulton School of Engineering
Arizona State University
Tempe, Az - 85281
pradyumna.kadambi@asu.edu

Wang, Yanyao.
Department of Computer Science
Ira A. Fulton School of Engineering
Arizona State University
Tempe, Az - 85281
ywang@asu.edu

Abstract

The rapid proliferation of information through digital media permits users to share content in a variety of forms, unlike past collaboration restricted by the information media. Furthermore, these interactions may now take place in real time, and without geographic restrictions. One key application in this domain includes Q&A forums, such as Stack Overflow, a member of the Stack Exchange community. This site provides a platform for users in various specialized fields to ask and answer questions within their area of expertise, and collaborate on problems. Question surfacing, an integral challenge to facilitating collaboration, is partially accomplished via post tagging, with one or more tags associated with a particular question. Users may self-tag their posts to categorize their content, and other users may browse new questions by category to narrow their collaboration scope within their relevant domain or sub-domain. However, it may not be obvious to new users (or even experienced ones) which tags might be appropriate for a given question. In this report, we present a method to automatically suggest appropriate tags based on the content of a post on-the-fly. Empirical results suggest that our

proposed ensemble network approach can effectively recommend relevant tags for a variety of posts, with a relatively high degree of accuracy.

Categories and Subject Descriptors

[**Information System**]: Social Recommendation, Social Tagging, Recommender systems

General Terms

Algorithms, Recommendation, Experiments, Performance, *on-the-fly* Recommendation.

Keywords

Stack Overflow, Stack Exchange, tag recommendation, data categorization, neural network, term-frequency, inverse-document frequency, histogram, ensemble.

1 Introduction

The massive explosion of information via digital media has spawned a critical need to better organize and deliver relevant content to participating users. Tagging, a process which associates a set of keywords with the content, has served as one component of these recommendation systems, to better organize content organization and delivery. Tagging is used in many domains, including image classification [6, 25], academic publications [18, 22], content bookmarking [29, 31], and more. In fact, the concept has even been extended from content-based to user-based tagging [32], also known as collaborative filtering, to better connect users with similar interests by leveraging individual histories and common backgrounds.

Figure 1: Top-6 tag suggestions containing the user typed tag as a substring - 'python'.

Prior applications usually address individual content tagging, rather than community-based, shared content such as that found on Stack Overflow (SOF); these types of content are driven solely by the users, and play a crucial role in facilitating cooperation and collaboration. SOF focuses on the

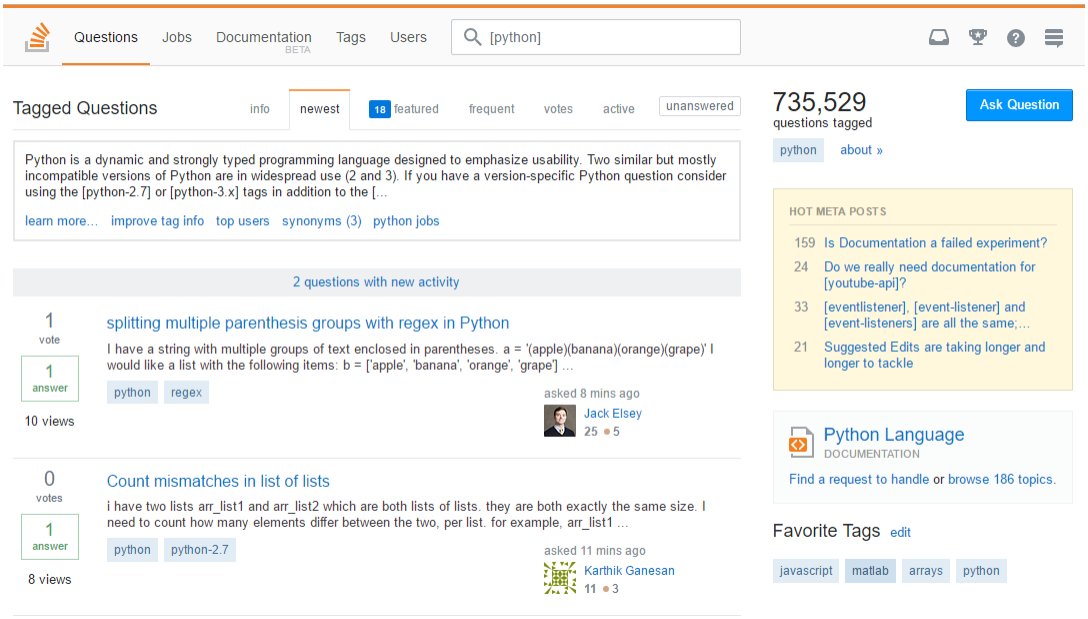

Figure 2: Search results based on the tag ‘python’.

subdomain of programming and software engineering knowledge[1], in the form a public question-and-answer (Q&A) sessions on user-defined subtopics.

SOF relies heavily on post metadata, including tags, for effective content discovery for site visitors. Tags are also used for indexing and searching content on the service, and are used to estimate the size of subcommunities within the broader SOF userbase. For example, in Figure 1, when a user enters “python” as a search term, SOF displays the top 6 tags that contain python as a substring, as well as showing the number of posts that contain the respective tag. Furthermore, in Figure 2, we see that when a user enters “python” as a search keyword, the site returns all posts tagged with python, or tags that contain python as a substring.

The scale of information in these services can easily outpace human capabilities of categorization, as they are aggregations of a large number of participants. Therefore, user error can easily occur due to lack of knowledge, or user confusion. By recommending tags based on prior knowledge (Figure 1), the integrity of the system can be enhanced by avoiding the introduction of tag synonyms, and aiding inexperienced users by recommending relevant tag suggestions directly from the user interface.

Finally, although these kinds of tag recommendation systems are not new, their development can still be improved via the application of novel machine learning techniques, which will subsequently produce richer and more effective tagging a categorization for user-driven content of all types.

2 Background and Motivation

Use of tags as an arsenal for content categorization is a very common in todays applications, specially in web-based Q&A forums. Tags are often considered as a type of metadata that best describes the content associated with it.

In the Figure 3, we can see a post with the associated user-defined tags. With the amount of content available with the post, we can describe post in a way that the user intends to determine the process to print all the elements organized in a multi-dimensional array created using the numpy package for python. In addition to this, based on the user’s key requirements, the associated tags are - python,

[1]Stack Overflow is one of 150+ community sites operated by Stack Exchange. SOF was launched nearly 8 years ago, and today contains 14M questions, 32M answers, with 7M users, with 8.1K new questions asked every day `http://stackexchange.com/sites?view=list#traffic`

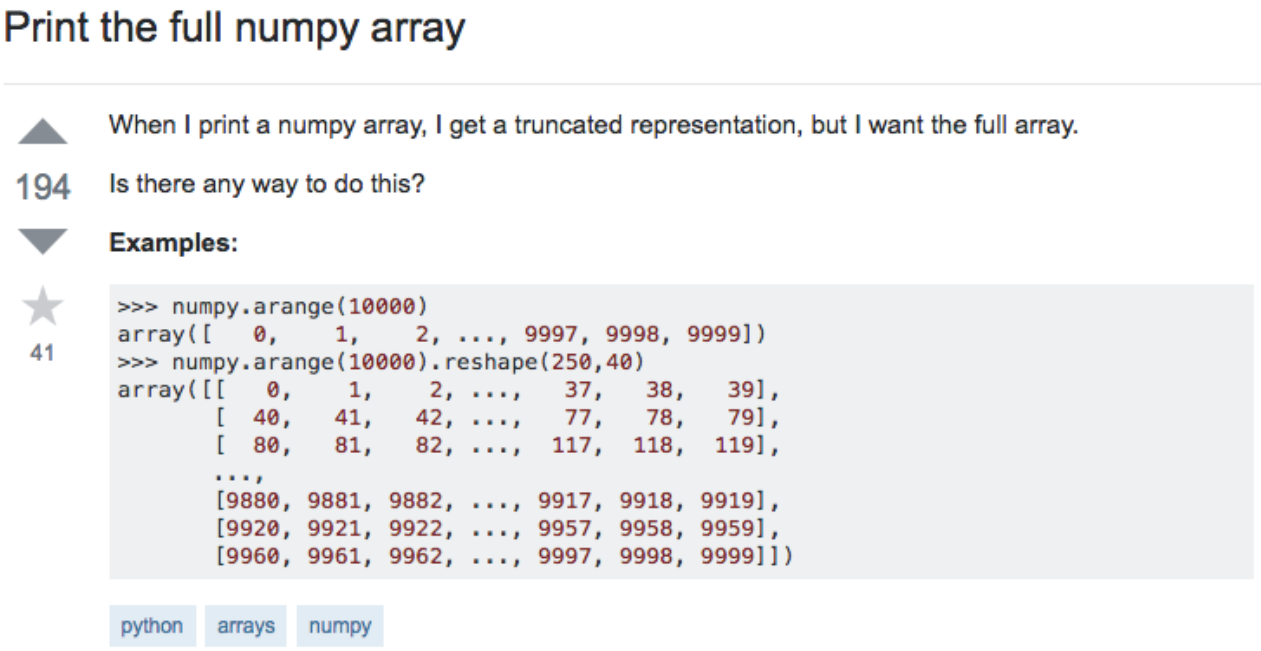

Figure 3: A sample post published on Stack Overflow - `http://stackoverflow.com/q/1987694/2962001`

array and numpy. Here, *python* is used to describe the programming environment, *numpy* places the role of describing the package used and finally, *array* is used to describe the component/aspect of the python package of interest for the user asking the post. It must be noted that, thought the contextual information is embedded in the post's title and body, the tag, mere three in count, enchants the entire story for a user with the perspective answer. Therefore, as noted earlier, tag are crucial for better content categorization to ensure proper propagation of the content to the desired audience.

2.1 *TagCombine*

In [30], Xia et al. proposed *TagCombine* - an automated tag recommendation system for information sites, such as SOF. The framework of *TagCombine* is organized under three step - first, multi-label learning component, second, similarity based tag ranking, and finally third, tag-term based raking components. This model is based on learning the tag associations in the historic data (terms in the posts). They combine the tag predicted from all three components and return the $top - k$ tags as specified by the user.

Tags associated with a post can be perceived as the label for the post. Particular in the case of SOF, tags are a multi-hot label, where a post can have more then one tag associated with them. Therefore, *TagCombine*, deploy a multi-label learning and ranking step to get the initial predictions for the tags(See Figure 4).

3 Existing Baselines

3.1 Support-Vector Machine – SVM

Support Vector Machine [13], SVM, is a supervised learning model that learning from the available data of the purposed of classification and regression. SVMs learns to define a hyperplane or a set of hyperplanes in the input space. The input space can be high- or infinite-dimensional space. These hyperplanes can be used in the tasks such as classification or regression. Hyperplanes aims to separate the two class of data and a a good separation is defined as a hyperplane that maximized the

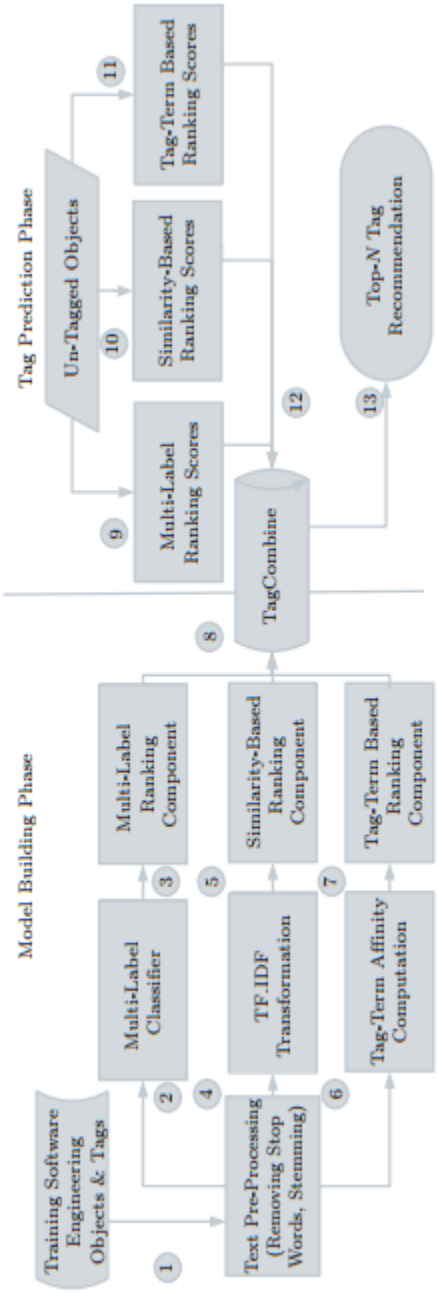

Figure 4: Tag Combine Framework

distance between the two class. This distance is measured between the nearest training samples of any class from the hyperplane, *functional margin*.

Given a training data, $(\vec{x_1}, y_1).....(\vec{x_n}, y_n)$, where $(\vec{x}, y)$ serves as an input pair where, $\vec{x}$ is the input vector and y is the label, where $y \epsilon (1, -1)$. Therefore, a hyperplane is defined as

$$\vec{w} \cdot \vec{x} - b = 0$$

, where $\vec{w}$ is a normal vector to the hyperplane and $\frac{b}{||w||}$ defines the offset(margin) from the origin.

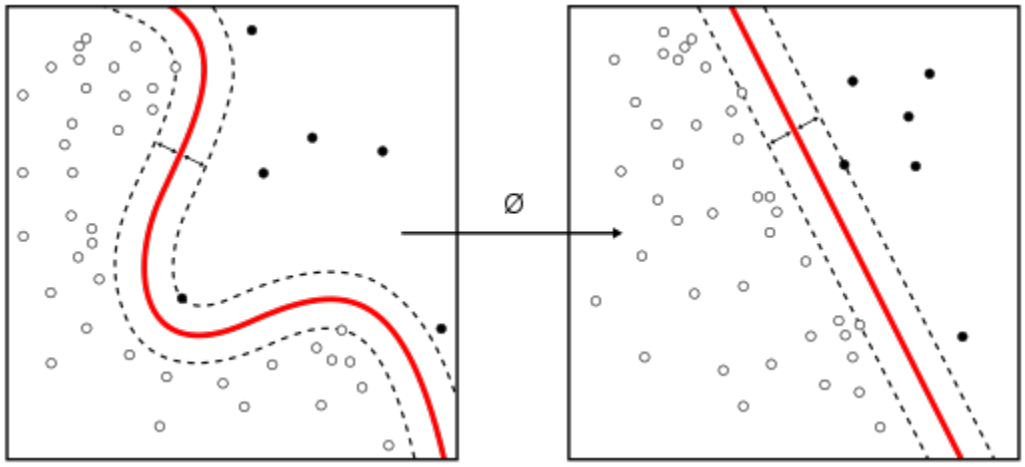

Figure 5: Source: `https://upload.wikimedia.org/wikipedia/commons/thumb/f/fe/Kernel_Machine.svg/512px-Kernel_Machine.svg.png`

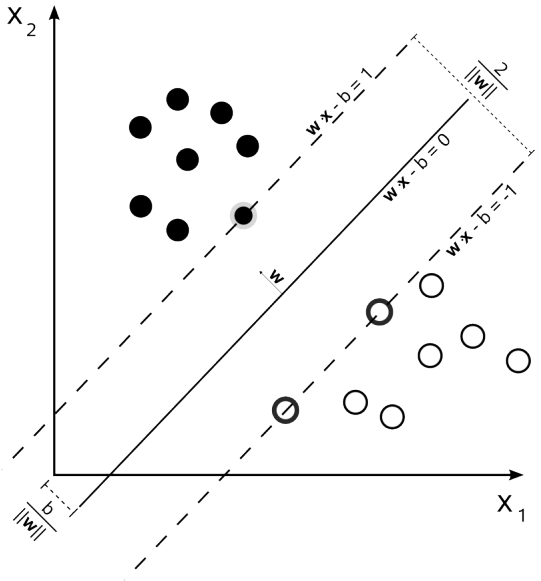

Figure 6: Source : `https://upload.wikimedia.org/wikipedia/commons/2/2a/Svm_max_sep_hyperplane_with_margin.png`

Furthermore, **hard margin** can be defined as

$$\vec{w} \cdot \vec{x} - b = 1 \; and \; \vec{w} \cdot \vec{x} - b = -1$$

and the margin is defined as $\frac{2}{||w||}$. It must be noted that the SVMs are binary classifiers and extending them for multi-class classification requires to implement a different class for every SVM.

3.2 Perception

While the naive implementation of Perceptron[13] is a supervised binary classification model, but it can be easily extended to carry out multi-class classification[14]. For simplicity of explanation we will focus on a binary perceptron.

Algebraically, perceptron can be defined as follows:

$$u(x) = \begin{cases} 1 & if \vec{w} \cdot \vec{a} + b > 0 \\ 0 & otherwise \end{cases} \tag{1}$$

where, $\vec{w} \cdot \vec{a} = \sum_{i=1}^{N} w_i \cdot x_i$, N is the dimensionality of input vector and b is the bias.

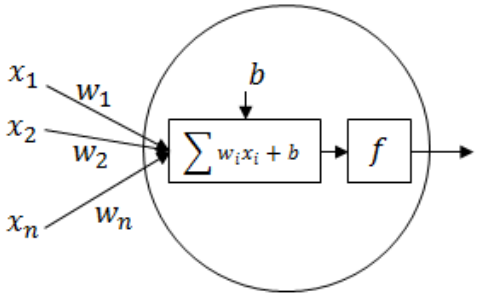

Figure 7: An abstract view of a neuron in a neural network, with bias b and activation function f

Perceptron Learning: Perceptron can be trained over a batch of data samples and the error can be correct to boost the classification accuracy.

1. **Step 1** - Predict the correct output
$$\hat{y} = f_t(x) = \vec{w_t} \cdot \vec{x} + b$$
2. **Step 2** - Error Correction
$$w_{t+1} = w_t + (y - \hat{y})\vec{x}$$

3.2.1 Convergence Condition

Convergence conditions plays a very crucial role in the training of perceptron. Convergence decides when the perceptron is fully trained and could be used to classify the unseen data. Also, Perceptron, binary classifier, is a linear model, therefore, it is never attain the state of correctly classifying all the data points in the case of non-linearly separable data.

As [9] states, perceptron can converge to a optimal solution when the data is linearly separable, however, this might be true if the data is not linearly separable. Therefore, the validation data is used to evaluation the model. Validation Classification error or validation loss is used as a metric to gauge the efficiency of the model.

3.2.2 Multi-Class Perceptron

As stated earlier, a binary perceptron could be generalized to a multi-class perceptron.

$$\hat{y} = argmax_y f(\vec{x}, y) \cdot \vec{w} \tag{2}$$

where, x and y are the input vector and the output label and $\vec{w}$ is the normal vector along the hyperplane. Furthermore, the learning step or the weight updation step remain similar to the naive case.

$$w_{t+1} = w_t + (f(\vec{x}, y) - f(\vec{x}, \hat{y}))$$

3.2.3 Activation Functions

Activation function is a function that defines the output of a node in a neural network for a given input or a set of input. Activation functions are perceived to resemble an abstract view of rate of action potential firing in a cell or node [24].

For the said purpose, reacting to stimuli, there are multiple activation function available:

– **Identity Function**[21] – Formally, an identity function, f, is defined as
$$f(x) = x$$
where, x is the input to the activation function. In simple words, identity function returns the same value as it's argument. Algebraically, $f : M \to N$ is a function that where domain of M and N is identical.

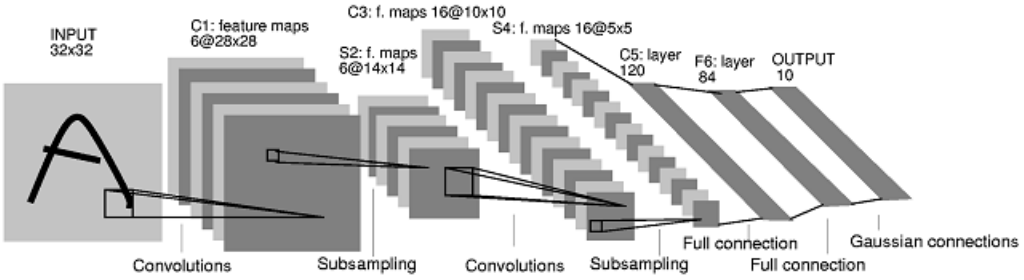

Figure 8: A detailed view of Convolutional Neural Network. Source: `https://i.stack.imgur.com/oUwMk.png`

– **Binary-step Function**[12] – This function is also known as Heaviside step function, denoted by $H(x)$.

$$H(x) = \begin{cases} 0 & x < 0 \\ 1 & otherwise \end{cases}$$

where, x an be assumed as the output of the neuron in the network. This function can be perceived as a discontinuous discrete function, whose value is 0 for negative inputs and 1 for positive inputs.

– **Rectified Linear Unit Function (ReLU)** is a special case of linear function or identity function, where

$$H(x) = max(0, x)$$

ReLU is also known as Ramp function, analogous to half-wave rectification function. ReLU were first given by [19], a publication in Nature Journal and was introduced in CNN in 2011 by [16].

– **softmax Function**[10], is a generalized logistic function that translated an n-dimensional input vector into an m-dimensional output vector of real values in the range of (0,1) which add up to 1.

$$softmax(\vec{x}) = \frac{e^{\vec{x}}}{\sum_{i=1}^{n} x_i}$$

These real values can be further considered as the probability of input vector being classified as either of the output label.

– **Sigmoid Function**, is a special case of logistic function defined as

$$sigmoid(\vec{x}) = \frac{1}{1 + e^{\vec{x}}}$$

, where the result real values can be perceived as the confidence of the input vector being classified as a particular class.

– **tanH Function**, is a rescaled sigmoid function with the output ranging between [-1,1] in contrast to the sigmoid's range of [0,1].

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

3.3 Convolutional Neural Network – CNN

Convolutional Neural Networks, CNN, are similar to MLP in terms of components involved in the construction of the networks that is neuron and the activation functions. However, their key

distinction is in terms of the connectivity between the adjacent neurons. That being said, CNN have n explicit assumption that the input data is a sequential data and the data can be an image, a time series or a voluminous input.

3.3.1 Components of CNN

- **Input Layers** - takes as an input the raw image files along with it's color channel (if any).

- **Convolution Layer** - This layer is responsible for the identifying the localized features in the input volume. This is achieved by defining a convolution kernel of a given window size with identical depth to that of the input layer. The kernel contains random real numbers between the range of (0,1) and the dot is taken with the local region in the image and the kernel is moved across the spatial dimensions in the image.

- **Pooling Layer** - Pooling layer is used as a down sizing measure to reduce the size of intermediate data that propagates towards the output layer.

- **Flatten** - While the input layer is a *two*-dimensional input where as the output layer has a *one*-dimensional input/output, therefore, it is required to translate the input to the compatible format to that of output layer. Thus, to achieve the said goal flatten layer is used.

- **Fully Connected Layer** - As the neuron at the convolution layers are flattened, they are connected to the output layer via a fully connected layer.

- **Dropout**, is used to reduce the over-fitting of network to the training data. Droput layer randomly pick a kernel to drop for different training sample batch and continues to train the network, asuuming that the kernel doesn't exist[28].

3.3.2 Spatial Arrangement

Spatial arrangement consists of mainly three hyper-parameter:

- **Depth** control the number of kernel in the output layer. Depth in other words is the number of kernel in the next layer.
- **Stride** is defined as the parameter that controls the movement of convolution kernel along the spatial dimension. With the use of strides, overlapping amongst the convolutional kernel can be controlled.
- **Zero-padding** control the type of data to pad, in the case of kernel overflowing the image when moved spatially to convolve the image or the input volume.

3.3.3 An Intuitive meaning of a CNN

In contrast to the conventional neural network that resemble a human brain in terms of neuron and how they are connected, CNN's key difference lies in the localized learnable filters. Each filter vary in spatial dimension (small than the input) but has equivalent depth (color the color channel). Each filter is slided over the entire spatial length of the input and the resultant convolution output is feed to the next layer after the activation function is applied.

4 Proposed Approach: Ensemble Learning

As observed in the Section 3, the key draw back of using SVM and Perceptron is the fact that they tend to fail with the increase in input and output dimensionality. With input dimensionality being the size of input space and the output dimensionality being referred to the number of labels or class that a classifier is required to learn. Understand the incumbent requirement of more intelligent models: we propose two families of models:

1. Multi-Layer Perceptron

2. Network Ensemble
   - Most-Voted Ensemble
   - Averaged Ensemble

4.1 Multi-Layer Perceptron – MLP

For learning the pattern of post associate with various tags we rely on a *simple multi-layer* perceptron (MLP) based neural network. We pick this simple neural network structure to better observe the impact of the proposed SOF post description method.

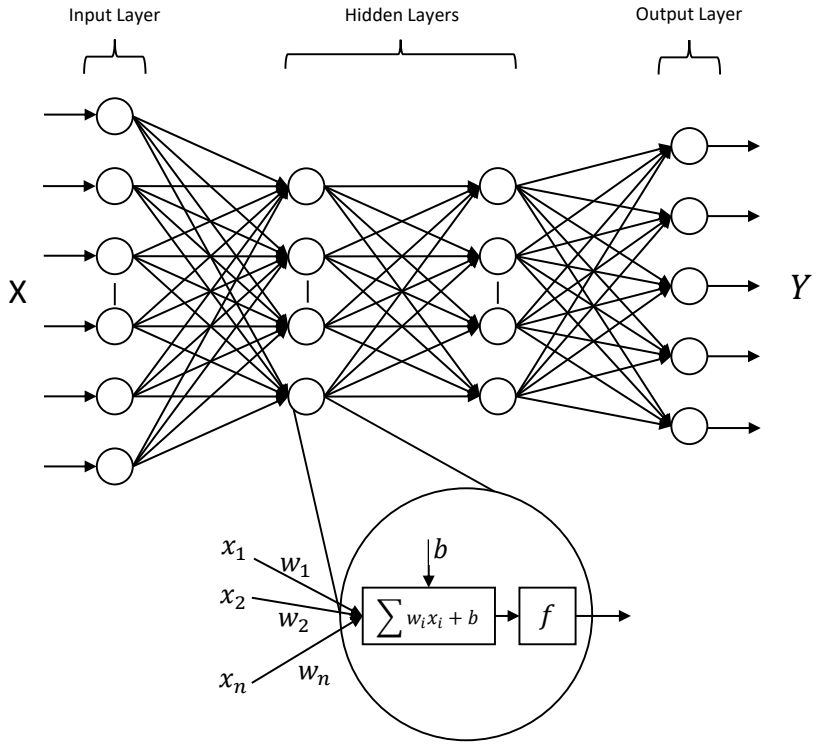

Figure 9: An abstract view of the multi-layer perceptron used for action learning using spatio-temporally localized features

MLP is a sequential neural net consisting of an input layer, one-or-more hidden layers, and an output layer (Figure 9). The weighted interconnections underlying the MLP allow us to learn a non-linear classifier to map inputs of the neural net to its outputs [9]. In general, the choice of hyper-parameters that define the topology of the underlying MLP network is both a crucial and difficult task. It is equally important to carefully select a set of activation functions, for the nodes of the network, and a training algorithm as these can have significant impacts on the performance of the network: improperly constructed MLPs may lead to over-fitting or under-fitting of data due to network induced noise.

For the choice of activation function, we conjecture that for the hidden layer, *rectified linear unit* (ReLU) and for the output layer *softmax* will help train a network to get better accuracy: ReLUs minimizes the loss of the gradient at the hidden layer and sigmoid allows probabilistic classification of training data during multi-class classification.

Note that, while the structure of the MLP is simple, it is important to account for a sufficient number of connections (neurons at hidden layer) to prevent over-fitting and non-convergence of the network[5]. Therefore, in the next section, we present tag prediction results for multiple network structures (i.e., depth and neuron densities).

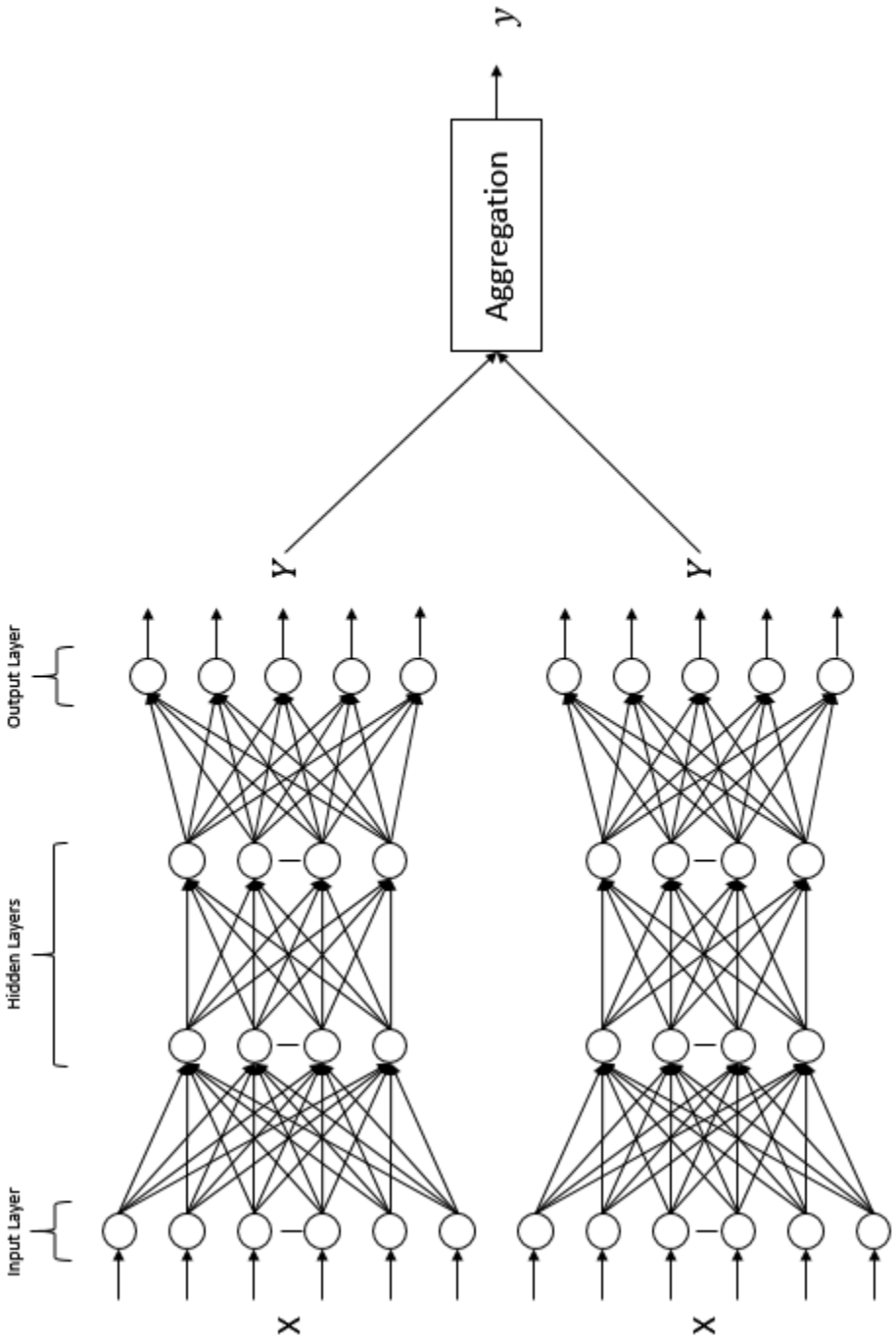

Figure 10: An abstract overview of an ensemble network with two models and with an aggregation function to combine the predict of individual models. This aggregation function can either be voting- or average-based.

4.2 Network Ensemble

While Multi-Layer Perceptron (MLP) are not new to the field, however, they have faced, and facing, sever criticism for their exponentially large hyper-parameter space. Due to the large parameter space it is difficult to find the best model to the task of interest. Therefore, Network Ensemble is a process of generating multiple models for a given model class and combining them to achieve the goal.

The way a set of models are combined into an ensemble is called aggregation and in this report we propose two types of aggregation mechanism: first, voting- and, second, averaged-based ensemble.

4.3 Most-Voted Ensemble

Voting-based classification, Maximum Voting Learners (MaVLs), has been there, as early as 1998 [7, 20]. Voting has shown promising results, bagging and AdaBoost when compared against state-of-the-art Decision Tree and Naïve Bayes, that have repeatedly proven to be optimal. Voting-based or Most Voted Ensemble is defined as, as follows. Let us assume a family of models

$$F\vec{x} = m_1(\vec{x}), m_2(\vec{x}), ..., m_k(\vec{x}) \tag{3}$$

where, each model,m, in the family, F, classifies the input,$\vec{x}$, independently.

$$\hat{y} = argmax_y m(\vec{x}) \tag{4}$$

Here, each model predicts a certain class for the input sample and the prediction with the best confidence or the highest probability is selected.

4.4 Averaged Ensemble

While it is not possible to pick the best models to define the network ensemble and the voting-based ensemble is susceptible to the noisy models as well. As the voting is based on the maximum confidence and it is observed that the noisy model tend to have significantly strong confidence in misclassification. Therefore, there is an inherent need to overcome the dominance of noisy models. With this aim, we propose the use of average function as the aggregation function to combine different models.

The use of average function is based on the underlying assumption that the family of models is not entirely composed of the noisy models, but the noisy models are rare, a handful.

Therefore, we define the classification function as

$$\hat{y} = avg_y m(\vec{x}) \tag{5}$$

We the averaging step, we aspect to see the dominance of intelligent models rather than the noisy ones. Given that the underlying assumption is that the noisy models are rare, thus their dominance is expected to be suppressed by averaging out.

5 Experiments

5.1 Data set

We use the anonymized version of the use-generated data on Stack Exchange Forum[1] - Stack Overflow[2]. The available dataset was available as zipped `XML` files. The data set can be accessed here [4]. Thought the compressed version of the dataset, `posts.xml`, is 15Gb, but the uncompressed version in 50 Gb with approx 33+ million posts (including questions and answers both). In general, *one-third* of the posts are questions and the remainder are the answers to the posts. The dataset contains, approximately, 47k tags[2]. Stack Exchange updates it's data quarterly.

5.1.1 Data Preprocessing

It is well known the data in raw state is the set of alpha-numeric characters from which information can be extracted. To facilitate the process of information extraction, it always preferred to extract

[2]The data is available under cc-by-sa 3.0 license.

set of words (text mining) that contribute to a distinctive and discriminative data representation. Therefore, the domain of information retrieval has long concentrated on find the components in text that are least useful when applications involve tasks, such as classification and clustering.

Stop Words[11, 26] - It is well known that some word in any language, per say English, has some redundant words that are very common that they have minimal discriminatory power in them. Therefore, to remove stop words we use `NLTK.corpus.stopwords` collection[3].

There are various types of stop-words:

- Determiners - Words preceding the nouns.
- Conjunctions - Words connecting nouns, phrases, and clauses.
- Prepositions - Words representing temporal and spatial relationships.

HTML Tags[17] - In the age of the digital world, data comes encoded inside HTML tags which is of not importance while performing classification. We use Regular Expression, `re` package, to remove <html> tags - `re.sub('<.*?>', '', post)`

Lower Case - To minimize the redundancy in the vocabulary, e.g. Test, TEST, test, we transform the entire posts to lower case typography. `post.lower()`

Non-Alpha Characters - We removed all non-alpha characters from the post, `re.sub(r'([^\s\w]|_)+', '', post)`

Spacing - Format the posts to have consistent spacing, replacing **tabs** with single *space* and also multi with single space,`re.sub(' +', '', post)`

5.2 Data Representation

Given the fact that the data generated at Stack Overflow is user-contributed, it is very rare, nearly impossible to maintain homogeneity.

5.2.1 Frequency Histogram

Hans Peter Luhn [23] was one of the first to propose term weighing based on the Luhn assumption, weight of a term is proportional to the term frequency in the document. Hans states that the Frequency Histogram or Term-Frequency histogram is represented by

$$tf(t', d) = \frac{f_{t',d}}{\sum_{t=1}^{T} f_{t,d}}. \tag{6}$$

where, f is the frequency of term, t is the term of interest and d is the document containing the term, t. Given that the content is user-contributed, it is very often that the post be of different length (word count), and contain different set of words. Therefore, normalizing the frequency of occurrence of term with the total number of terms in the documents helps overcome the difference in number of total terms across different documents.

5.2.2 Term Frequency, Inverse Document Frequency Histogram

While the term frequency is driven by frequency of occurrence, it is very often that the use of frequency might mislead, when the words are very common, in a document or in a collection of documents as a whole. In 1972, Karen Sparck Jones [27] proposed, what is called today - inverse document frequency - and became the corner stone of term weighing in the field of text mining and information retrieval.

$$idf(t, D) = log \frac{N}{|\{d \, \epsilon \, D : t \, \epsilon \, d\}|} \tag{7}$$

where, N is the total number of documents in the databases, d is the total number of documents containing the term t. In contrast to the term frequency where each term is considered of equal importance, inverse document frequency gives more importance to the terms that are rare.

Therefore, the tf-idf can be defined as:

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D). \tag{8}$$

5.3 Tag Synonymization

The number of unique tags indexed by SOF, as of December 2016, are approximately 47k, in contrast to the 38k reported by Beyer in [8]. It is important to use precise tags for information categorization, however, the need for precision has lead to an increase in the number of tags used. This can brought upon the urgent need to downsize the tag space, thus finding clusters of tags that could be represented by a single, may be simple, tag.

5.4 Evaluation

It is important to understand the distribution of data set that we are considering for the setup of the evaluation. Given the size of the data it nearly impossible to work with the entire data without can transformation.

We first, the down-size the output label space, by carrying out tag clustering based on the synonymization. Example: replace 'python', 'python2.7' and 'python3.4' with simple python. Furthermore, we observe that the top-20 (by frequency count) covers the 85% of the data(see Figure **??**).

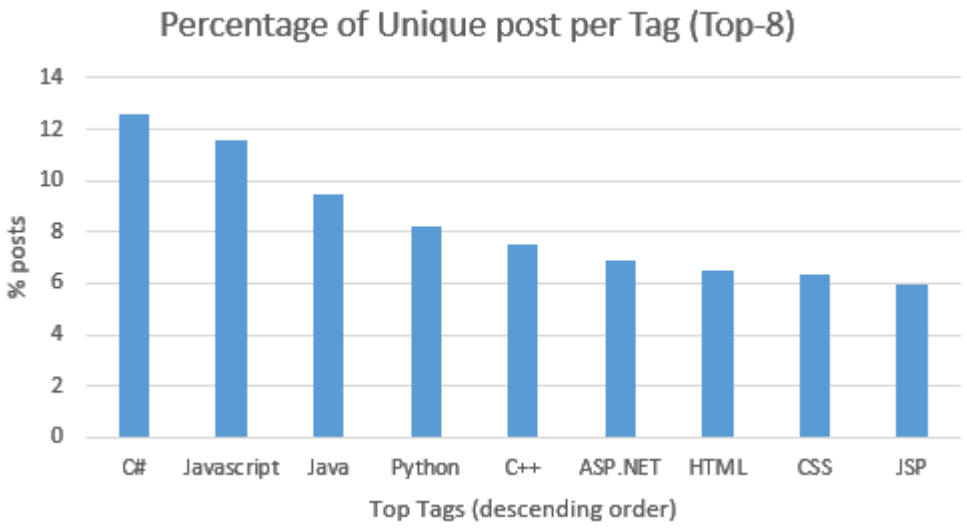

Figure 11: Top-10 tags with percentage of unique posts containing either one

5.4.1 Classification Accuracy

It can be clearly seen in the Figure 12, that the average-ensemble outperform other models. Reason being the use of multiple model, model family, that are non-redundant classifiers and provides an additive classification of the test data. Whereas in the case of vote-based ensemble, the noisy models tends to dominate the classification results thus bringing down the overall accuracy results.

5.4.2 Classification for different Training size

While computing this accuracy results, it was made sure that the data is unbiased towards any class and there is a balance amount of training data for each tag. Furthermore, randomly varied percentage of training data was selected and the average of 5 random runs is reported (See Figure 13. The balance in training data is crucial as the baseline data contain different proportion of posts (training samples) for different tags 5.4

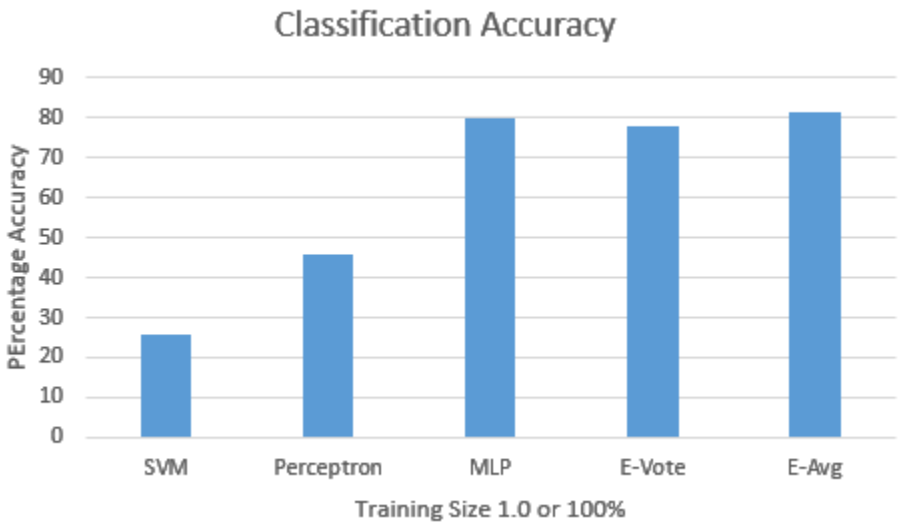

Figure 12: Classification Accuracy when entire training data is used to train the models.

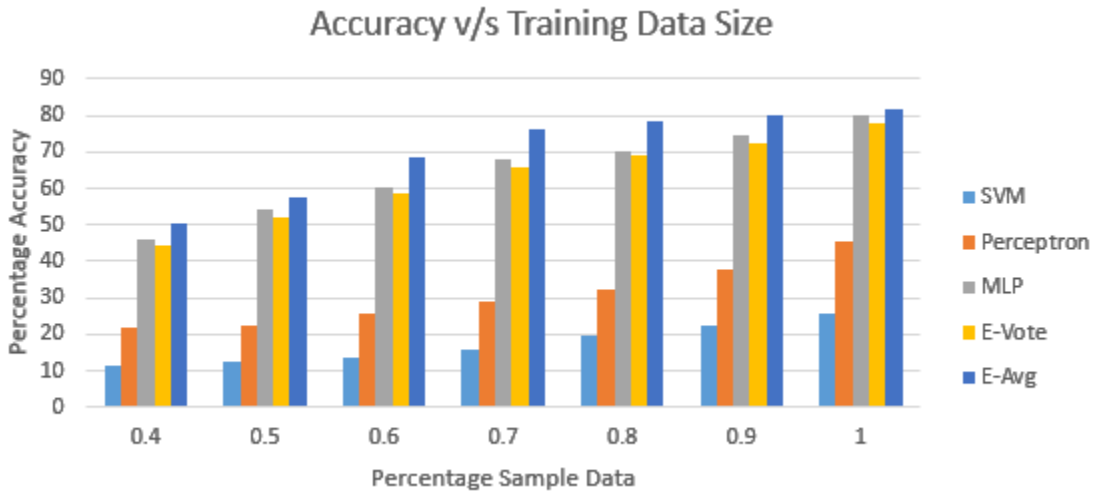

Figure 13: Classification Accuracy of Different models with different training data size.

6 System Configuration

6.1 Setting up the System

The following setup instructions are designed to run our project in Windows environment. The source-code is available online[15].

6.1.1 Set up IDE

- Download Eclipse. Prefer the latest release for Java EE Developers)
- To get python working go to, Help > Eclipse Marketplace > Find: PyDev > Install (if not installed)
- Connecting Github to Eclipse, Help > Eclipse Marketplace > Find: Egit > Install (if not installed)

6.1.2 Set up MySQL

- Download latest MySQL Community Server, prefer downloading the installer instead of the .zip file, installer configures everything by itself.
- Give your preferred username and password, or you can set default **root** for both, once done, update the variables `MYSQL_USER_NAME` and `MYSQL_PASS_WORD` in `MySqlPkg.py`.
- **Set up MySQL Python Driver**: Run `conda install -c anaconda mysql-connector-python=2.0.4`

6.1.3 Set up Libraries

While there are many ways to setup python and associated libraries, the examples being individual or batch configuration. To do batch configuration, to setup Python 2.7, Numpy, Scipy, Keras, Tensorflow/Theano follow the following steps.

1. Install TDM FCC x64.
2. Install Anaconda x64 for **Python 2.7**.
3. Open Anaconda prompt.
4. Run `conda update conda`
5. Run `conda update --all`
6. Run `conda install mingw libpython`

6.1.4 Anaconda as default python interpreter

1. Right click `My Computers` or `This PC` to open `Properties` and click on `Advanced System Settings` on left sidebar menu to open `System Properties`.
2. In `System Properties` click on `Advanced` tab, click `Environment Variables`, add new or edit `PATH User Variables` and add the location of Anaconda on your machine (eg. `C:\Anaconda2`).
3. In `System Variables`, edit or add new `Path` variable and add the following paths - `C:\Anaconda2` and `C:\Anaconda2\Scripts` and `C:\Anaconda2\Library\bin`. the path must resemble the location on your machine.

6.1.5 Set up `pip`

1. Download `pip`, get-pip.py
2. Run `python getpip.py`

6.1.6 Set up `Keras`

7. Install Theano, `pip install git+git://github.com/Theano/Theano.git` or `pip install theano`
8. Run `pip install git+git://github.com/fchollet/keras.git` or `pip install keras`

In the case of old keras version being install, consider running the command - `pip install keras --upgrade`.

6.1.7 MacOS

In the case of MacOS or OS X, it is observed that the default backend is tensorflow, whoch any be easily modify to theano by adding the `export KERAS_BACKEND=theano` in default environment variable.

6.2 Loading the Data

Given the data is available in **XML** format of size `16GB` posts from from Stack Overflow. Thus making it impossible to load the entire data into the main memory and train a deep network. Therefore, it is incumbent that a disk-based read is performed and the data is loaded into a more manageable data structure.

6.2.1 Reading Posts

Based on the above observation we implement the package, `LoadData` which contains a method defined as `parse_post_data()` that can perform the above said job. The method, `parse_post_data()` take the physical file location, on disk, as argument and iteratively loads posts - one at a time.

Load Data - Set the `Config.DATAPATH` variable in `Defaults` package to have the physical file location of **posts.xml**. Once the variable name is updated, simply run the `ReadData.py`

6.2.2 Cleaning Posts

Stop Words - It is well known that some word in any language, per say English, has some redundant words that are very common that they have minimal discriminatory power in them. Therefore, to remove stop words we use `NLTK.corpus.stopwords` collection.

HTML Tags - In the age of the digital world, data comes encoded inside HTML tags which is of not importance while performing classification. We use Regular Expression, `re` package, to remove `<html>` tags - `re.sub('<.*?>', ' ', post)`

Lower Case - To minimize the redundancy in the vocabulary, e.g. Test, TEST, test, we transform the entire posts to lower case typography. `post.lower()`

Non-Alpha characters - We removed all non-alpha characters from the post, `re.sub(r'([^\s\w]|_)+', ' ', post)`

Spacing - Format the posts to have consistent spacing, replacing **tabs** with single *space* and also multi with single space, `re.sub(' +', ' ', post)`

6.3 Data Representation

Histogram, in general, are used to define the distribution of data. In our case, the distribution is defined in terms of the number of times a word occurs in a post. To extract the various histogram representation for our data, we have created a package `WordExtractor`.

This package is designed to extract various types of histograms from the posts stored in the MySQL using the `LoadData.ReadData`.

Types of Histogram:

- Naive Frequency Histogram - `naive()`
- TF or Normalized Frequency Histogram - `tf()`
- TF-IDF Frequency Histogram - `tfidf()`

6.3.1 Assumptions

- The JSON array contains all the posts stored in the `questions` table or the table name specified in `Defaults.MySqlPkg.MYSQL_QUESTION_TABLE_NAME_CLEAN`, unless a limit is specified.
- Here it is assumed that the data is free from stop words.

6.3.2 Output

Returns a `JSON` array of the form `[{...},{...},....,{...}]` to the disk, where each JSON `object` in the `array` is in the format `{id:postid,`

`words:[w1:f1,w2:f2,...,wn:fn],tags:[t1,t2,...,t5]}` and a JSON `object` containing the metadata of the words in the entire data corpus.

6.4 Naive Frequency Histogram - `naive()`

6.4.1 Specification

`naive()` counts the frequency of each word in the post. Ex. consider a sentence, *Cat ran after another Cat*, therefore the output would be {`cat:2,ran:1,after:1,another:1`}

6.5 TR or Term Frequency Histogram - `tf()`

TF (Term Frequency) Histogram, `tf()`, works identical to `naive()`, but the returned frequency for each word in a post is normalized by the total number of words in the post.

6.5.1 Specification

`naive()` counts the frequency of each word in the post. Ex. consider a sentence, *Cat ran after another Cat*, therefore the output would be {`cat:0.4,ran:0.2,after:0.2,another:0.2`}

6.6 TF-IDF or Term Frequency & Inverse Document Frequency Histogram - `tfidf()`

TF-IDF (Term Frequency & Inverse Document Frequency) Histogram, `tfidf()`, works identically to `tf()`, but the returned term frequencies for each word in a post are weighed against the `log(Total number of documents/Number of documents containing the word)`.

6.6.1 Specification

Consider the sentence *Cat ran after another Cat*, therefore the frequency histogram would be {`cat:2,ran:1,after:1,another:1`} and the corresponding term-frequency histogram would be {`cat:0.4,ran:0.2,after:0.2,another:0.2`}. Now consider there are **1000 documents** of which **10 documents** contains the word **cat** in them. Therefore the `tfidf` value for the word cat would be `0.4*log(1000/10) = 0.8`

6.7 Modeling & Classification

This page documents the `MyModel` package in use to train and test different StackOverflow posts. Here we will define two types of models: `mlp` and `cnn`. The `Model` package uses the `WordExtractor` package to get the desired inputs and the output.

6.7.1 JSON to Numpy - `getdata()`

The **Keras** library is designed to take input and produce outputs in form of a numpy array, therefore, we designed the method, `getdata()` that takes as input the `json` string containing histogram and the metadata and returns corresponding training data & label matrix and testing data & label matrix.

6.7.2 Training the Model

Once, the matrix representation is generated using the `getdata()` methods, use either of the model methods, `mlp()`, `en_avg()`, `en_pool()`, defined in `MyModel`. These methods are

7 Conclusion

Although state-of-the-art methods using SVM and Perceptrons may be optimal, they suffer from two critical limitations. First, both are linear classifiers, which limits their practical application as the size of the data set scales. Second, they cannot easily learn complex patterns, such as the distribution of words.

A multi-layer perceptron approach overcomes these limitations, but due to the exponentially-large hyperparameter space, it becomes difficult to find the best network configuration. Additionally, improper parameter choice may lead to the network being under- or over-fit, reducing accuracy.

To address these challenges, multiple models can be leveraged to determine a set of non-redundant, complementary multi-layer perceptron configurations. This so-called family of models is referred to as a Network Ensemble, and the learning process by which it is formed is called Ensemble Learning. We have shown that with the use of ensemble aggregation via averaging, we can achieve a gain in classification accuracy of approximate 2%; we postulate that these gains can be credited to the additive nature of our model, with each constituent model learning patterns differently.

Acknowledgments

We would like to thank Professor Tong for providing us with the opportunity to explore the domain of deep networks for tag recommendation, our TA Boxin Du, and the various proctors and guests for their constructive and positive feedback.

References

[1] Stack exchange. `http://stackexchange.com/`, 2008.

[2] Stack overflow. `http://stackoverflow.com/`, 2008.

[3] Natural language toolkit. `http://www.nltk.org/`, 2009.

[4] Stack exchange data dump. `https://archive.org/details/stackexchange`, 2016.

[5] Driss Agliz, Abderrahman Atmani, et al. Seismic signal classification using multi-layer perceptron neural network. *ICCA*, 79(15), 2013.

[6] Ashton Anderson and Adam Vogel. Tagez: Flickr tag recommendation. 2008.

[7] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine learning*, 36(1):2, 1998.

[8] Stefanie Beyer and Martin Pinzger. Synonym suggestion for tags on stack overflow. In *Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension*, pages 94–103. IEEE Press, 2015.

[9] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

[10] Christopher M Bishop. Pattern recognition. *Machine Learning*, 128:1–58, 2006.

[11] Hua Bolin. Stop-word processing technique in knowledge extraction. *New Technology of Library and Information Service*, 8:48–51, 2007.

[12] Ronald Newbold Bracewell and Ronald N Bracewell. *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York, 1986.

[13] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[14] Yoav Freund and Robert E Schapire. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296, 1999.

[15] Yash Garg, Hans Behrens, Pradhyumna Kadambi, and Yanyao Wang. QTagger. `https://github.com/yashgarg1232/QTagger/`, 2017.

[16] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. 2011.

[17] Ian S Graham. *The HTML sourcebook*. John Wiley & Sons, Inc., 1995.

[18] Yaakov HaCohen-Kerner, Zuriel Gross, and Asaf Masa. Automatic extraction and learning of keyphrases from scientific articles. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 657–669. Springer, 2005.

[19] Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947–951, 2000.

[20] Gareth James. *Majority vote classifiers: theory and applications*. PhD thesis, Stanford University, 1998.

[21] Anthony W Knapp. *Basic algebra*. Springer Science & Business Media, 2006.

[22] Michał Łopuszyński and Łukasz Bolikowski. Towards robust tags for scientific publications from natural language processing tools and wikipedia. *International Journal on Digital Libraries*, 16(1):25–36, 2015.

[23] Hans Peter Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development*, 1(4):309–317, 1957.

[24] Jesper Lützen. Heaviside's operational calculus and the attempts to rigorise it. *Archive for History of Exact Sciences*, 21(2):161–200, 1979.

[25] Börkur Sigurbjörnsson and Roelof Van Zwol. Flickr tag recommendation based on collective knowledge. In *Proceedings of the 17th International Conference on World Wide Web*, pages 327–336. ACM, 2008.

[26] Catarina Silva and Bernardete Ribeiro. The importance of stop word removal on recall values in text categorization. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 3, pages 1661–1666. IEEE, 2003.

[27] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

[28] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[29] Lubomira Stoilova, Todd Holloway, Ben Markines, Ana G Maguitman, and Filippo Menczer. Givealink: mining a semantic network of bookmarks for web search and recommendation. In *Proceedings of the 3rd international workshop on Link discovery*, pages 66–73. ACM, 2005.

[30] Xin Xia, David Lo, Xinyu Wang, and Bo Zhou. Tag recommendation in software information sites. In *Mining Software Repositories (MSR), 2013 10th IEEE Working Conference on*, pages 287–296. IEEE, 2013.

[31] Takumi Yoshida and Ushio Inoue. Bookmark recommendation in social bookmarking services using wikipedia. In *Computer and Information Science (ICIS), 2013 IEEE/ACIS 12th International Conference on*, pages 551–552. IEEE, 2013.

[32] Yuan Zhang, Ning Zhang, and Jie Tang. A collaborative filtering tag recommendation system based on graph.