STEVENS INSTITUTE OF TECHNOLOGY

**NAMES:** Xianfei Gu, Yang Yang, Rush Kirubi, Jianjie Gao, Praveen Thinagarajan

**COURSE:** BIA 660 (Web Analytics)

**TASK:** Group Project Report

**INSTRUCTOR:** Zachary Wentzell

**SEMESTER:** Spring 2017

## Problem Statement

Tagging involves using discrete descriptive words to label a document. In a world typically characterized by information overload from, most notably, the internet, it serves to help separate the wheat from the chaff. StackOverFlow and Quora are both rich question and answer sites that use tags extensively, with the former focused on computer programming queries and the latter, for general usage.

When statements or questions are rightly tagged, they facilitate easy retrieval of content. It's commonplace for the task to be handled by humans. While accuracy in doing so may be high, it's rather laborious and would benefit from automation. Focusing on StackOverFlow, we aim to investigate ways to tag questions fast. Specifically, we intend to use supervised machine learning to create a more dependable solution to this problem.

## Introduction

To maintain a consistent strategy in addressing the problem, we sought to employ the Cross-Industry Standard Process for Data Mining (CRISP-DM) [3]. CRISP-DM is an iterative methodology whose steps communicate clear objectives to domain and non-domain machine learning specialists alike. We undertook five of the six phases:

1. Business Understanding
2. Data Understanding
3. Data Preparation
4. Modelling
5. Evaluation
6. Deployment

As seen later, the results of the Evaluation stage was not fruitful enough to proceed with deployment.

## Business Understanding

The StackOverflow 2016 Developer Survey mentions that every 8 seconds or so, a question is asked on the Stack Overflow website [9]. Simple arithmetic shows us that this translates to 3,942,500 questions asked annually. And these are just questions related to coding.

Tagging, as observed in blogs and other online arenas, is a human task. A person scans content and from prior knowledge, which is recalling similar content, can label a piece of text manually. One possible solution is using semi-automated services such as Amazon's Mechanical Turk [4] which is about writing code on one end as a conduit that shows people questions and they respond accordingly with the appropriate tags. There are two downsides to this: First, while the service allows the requester to decide how much to charge, should we approximate that for every single question tagged we pay a cent, then

annually this is $39,425. Obviously, this an optimistic view, recalling that there's a backlog of questions not appropriately tagged and the amount approximated so far, is below the minimum wage acceptable to human workers who may not take it up. Second, this solution is still predicated on humans who do not make errors and know the subject matter thoroughly, which a problematic assumption to hold.
We are trying to investigate if Machine Learning yields a more efficient result. However, the scope of the project does not involve evaluating costs.

## Data Understanding

Stack Exchange provides the latest StackOverFlow questions regularly. We used 'stackoverflow.com-Posts.7z' [8] which contains StackOverFlow data that is in the eXtensible Markup Language (XML) format. We chose to select the most recent occurring questions that number about 4,500. On the other hand, Stack Overflow also provides a public application programming interface (API) [7] whose daily request limit is capped at 10,000. Given the readily accessible *Posts.xml* file, we proceeded to do work with this first and for continued testing, we will use data retrieved from the API.
The posts data consists of 11 features namely, Post Type ID, Parent ID, Creation Date, Score of the post, View counts, Body of the post, Owner User ID, Title, Accepted Answer ID and Comment Counts. *Figure 1* is a snapshot of a single question as organized within the greater Posts.xml file. For our natural language processing assignment, we consider only the *Title* and *Body*. With this, we generated a list of questions (combined title and body) and their corresponding tags.

```
  <row Id="41600586" PostTypeId="1" AcceptedAnswerId="41600788" CreationDate="2017-01-11T20:59:40.120" Score="0" Vi
ewCount="17" Body="&lt;p&gt;I'm looking at the docs for a command line tool, and in the section indicating options
there is: &lt;/p&gt;&#xA;&#xA;&lt;blockquote&gt;&#xA;  &lt;p&gt;--viewport-size &amp;lt;&gt;              Set viewp
ort size if you have custom&#xA;                                    scrollbars or css attribute overflow to&#xA;
emulate window size&lt;/p&gt;&#xA;&lt;/blockquote&gt;&#xA;&#xA;&lt;p&gt;Just wondering how this viewport-size is su
pposed to be passed. Just width in pixels? Two arguments for width and height of the viewport in pixels? I'm experi
menting with it but the results are somewhat confusing. For reference &lt;a href=&quot;http://wkhtmltopdf.org/usage
/wkhtmltopdf.txt&quot; rel=&quot;nofollow noreferrer&quot;&gt;here&lt;/a&gt; is the documentation page. &lt;/p&gt;&
#xA;&#xA;&lt;p&gt;Thanks!&lt;/p&gt;&#xA;" OwnerUserId="6263989" LastEditorUserId="6862601" LastEditDate="2017-01-11
T21:23:25.013" LastActivityDate="2017-01-11T21:23:25.013" Title="How is viewport size typically passed as an argume
nt?" Tags="&lt;html&gt;&lt;web&gt;&lt;command-line&gt;" AnswerCount="1" CommentCount="0" />
```

Figure-1: *Representation of a line in the StackOverFlow Posts.XML file*

## Data Preparation:

The first step in pre-processing the data is merging the content of the *title* and *body* attributes. Each post is thus represented by each merge. In what is generally described as text normalization, we proceeded to remove 'noise' and convert the data into a malleable form [2]. This begins with expanding contractions and then annotating text tokens with parts of speech tags which aids lemmatization. The goal of lemmatization is to reduce

each word to its dictionary base form. Later, we remove special characters stop-words and numbers as they hold no significant informational value.

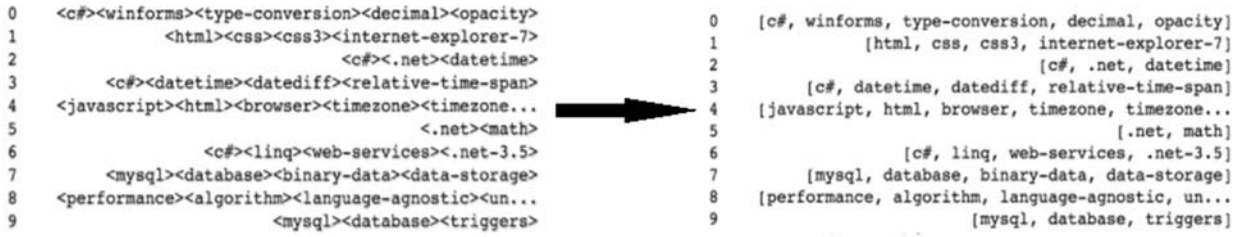


Figure-2: *List representation of Tags for each post*

Going forward, one route needed us to make separate columns for each of the tags we wanted to predict e.g. *is_java* is a binary 1 or 0 dependent on whether Java appears. We did this for the top 10 most frequently occurring tags. This was followed by converting the features to vector representations using the *Word2Vec* algorithm provided by the Gensim Python package.

## Modelling

*Related Current Methods:*

Addressing the problem relies on using natural language processing tasks, to derived numerical features that can be fed to a machine learning model. Of the mainstream approaches to solve natural language processing, is inferring semantics of a language. In English, the roughly eight parts of speech, are the building blocks of parsing prose. [6] Hidden Markov models have been a default option in solving the derivation of parts of speech. Although, the model is useful in predicting probable patterns, tagging content is concerned with annotation which is scarcely about discerning sequential patterns.

Topic modelling algorithms are contemporary methods of annotating large collections of documents. These techniques are unsupervised as they do not require a prior annotation of documents to infer themes, rather, they uncover the underlying themes from the content of the documents under review. [5] Latent Dirichlet Allocation (LDA) is an example of a topic modelling algorithm. It's termed 'latent', because it reveals hidden topics, or words that go together when inspecting a set of documents. The name 'Dirichlet' hints at the fact that it uses a probabilistic model, and specifically an assumption of a Dirichlet prior distribution. LDA assumes that the words that created a document in the corpus arose from a generative process. One can consider the process to be in three stages: a random selection of a distribution of topics, then randomly choosing a topic from the selected distribution before finally choosing a word from the vocabulary of this chosen topic.

*LDA and SVM*

We constructed an LDA model based on 10 topics. The output was 10-topic vectors over all observations (documents) that was fed into our Support Vector Machine classifier. Since we are working with the top 10 tags, we created 10 binary classifiers using this approach.

*Word2Vec and XGBoost*

This approach involved getting weighted vector representations of each document throughout the corpus using gensim. But this time, if a tag appeared in the document, we weighted this more heavily, as we posited that the tag that showed up as a word feature, was likely to be the tag being predicted. Next, we applied the eXtreme Gradient Boosting (XGBoost) library wrapped inside a One-Vs-Rest Classifier object from the Scikit-Learn library. One-Vs-Rest allows making multi-class predictions using one classifier as opposed to the 10 binary classifiers we built previously. Essentially, at prediction time, all classes are assigned a probability and the one with the highest probability is the predicted label. In addition, the hyperparameters below informed our Random Search Cross Validation routine:

```
param_distribs = {
       'estimator__kernel':[ 'linear', 'poly', 'rbf', 'sigmoid'],
       'estimator__C' : list(np.linspace(0.01,1,20)),
       "estimator__degree": [1, 2, 3, 4],
       "estimator__class_weight":['balanced'],
       "estimator__probability":[True]
   }
```

## Evaluation

Accuracy and Recall are germane metrics we use to evaluate our model. Simply put, accuracy is what we got right (true positives and true negatives) over the total number of observations. And of the observations that are relevant recall is the proportion our classifier was able to detect (true positives over true positives plus false negatives).

*LDA and SVM*

Of the 10 binary models created, the best one yielded an accuracy rate of 80% and a recall rate of 70%.

*Word2Vec and XGBoost*

After we got the best single model, we trained it on the whole training set and evaluated it on the test set. For our best single XGBoost model, we got an accuracy rate of 75% and recall rate of 64%. For every single run, we may get slightly different results because we were using random  search cross validation.

## Future Work

To get better results, we can further reduce noise by only using words that have more predictive power. Specifically, for each tag, we can split the target feature into two parts. eg.(*is_python*, *not_python*) and calculate the ratio, which is the frequency of each of the tokens in the *is_python* corpus divided by frequency of each of the tokens *not_python* corpus. Then we sort the ratios and only used those words with higher ratios based on a set threshold.

## Bibliography


[1] A. Géron, Hands-on machine learning with Scikit-Learn and TensorFlow.: O'reilly, 2017

[2] D. Sarkar, Text Analytics with Python. : Apress, 2017.

[3] P. Chapman, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer and R. Wirth, "CRISP-DM 1.0", The Modeling Agency, 2017. [Online]. Available: https://www.the-modeling-agency.com/crisp-dm.pdf. [Accessed: 09- May- 2017].

[4] "Pricing | Requester | Amazon Mechanical Turk". Requester.mturk.com. N.p., 2017. Web. 9 May 2017.

[5] "Probabilistic Topic Models", Computer Science Department - Columbia University, 2017. [Online]. Available: http://www.cs.columbia.edu/~blei/papers/Blei2012.pdf. [Accessed: 28- Apr- 2017].

[6] "Robust Part-Of-Speech Tagging Using A Hidden Markov Model - Sciencedirect". Sciencedirect.com. N.p., 2017. Web. 28 Apr. 2017.

[7] "Stack Exchange API", Api.stackexchange.com, 2017. [Online]. Available: https://api.stackexchange.com/. [Accessed: 02- May- 2017].

[8] "Stack Exchange Data Dump : Stack Exchange, Inc. : Free Download & Streaming : Internet Archive", Internet Archive, 2017. [Online]. Available: https://archive.org/details/stackexchange. [Accessed: 31- March- 2017].

[9] "Stack Overflow Developer Survey 2016 Results", Stack Overflow, 2017. [Online]. Available: https://insights.stackoverflow.com/survey/2016. [Accessed: 09- May- 2017]