

# Stack Exchange: Multi-Label Tag Classification

Christina Chang

Jonathan Quach

Troy Lui

## Introduction

### *Overview*

Stack Exchange is a network of questions and answers, which are asked and answered by users with Stack Exchange accounts. Each post starts out as a question asked by a user, and other users comment solutions or solution hints on the post regarding the question. Within each post, tags, or words/phrases that categorize a post under specific topics, are attached to the bottom of each question. With this in mind, we want to use Stack Exchange data, provided by a competition on Kaggle, to train models through neural networks and various classification algorithms using different types of methods to be able to classify posts with tags based on a question's title and text content.

Our motivation towards covering a classification problem is that even though it's such a large aspect within data science, we had little to no experience with classification. Therefore, we wanted to deal with a classification problem to broaden our knowledge to better prepare ourselves for inevitable classification problems in the future.

**Project Goal:** Classify select tags to StackExchange posts based on title & text content

After all algorithms and methods were conducted, we found that neural networks was the best way to tackle the multi-label tag classification problem. Specifically, the various categories' F-scores ranged from (0.2204 ~ 0.5286) and only lost out to binary relevance via decision trees in the robotics category (0.2204 vs. 0.3177). When combining all categories to create a comprehensive frame, neural networks also performed the best with an F-score of 0.4915. [Individual values viewable in results section]

## Methodology

### *Dataset*

The Stack Exchange dataset was received from a completed competition on Kaggle. It includes multiple datasets filled with posts that are split into categories regarding biology, cooking, cryptography, "do it yourself", robotics, and test (a physics data set). Each data set has four columns, which are id, title, content, and tags (with exception to the testing dataset, which doesn't have tags). Posts are given ids, which are unique numbers (1,...,n) where n is equal to the total number of posts within the dataset. The title and content columns contain the text of each individual post. Lastly, the tags column contains words or phrases that classify the post as pertaining to specific subjects. To note, posts can contain multiple tags.

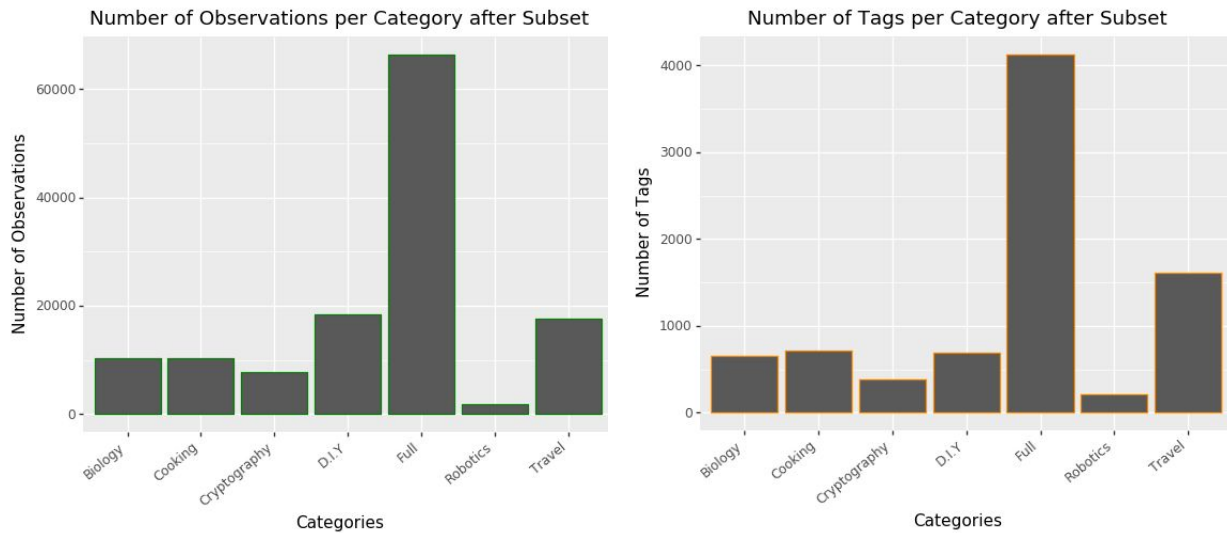
## Data Cleaning

To clean the data, we removed angle brackets and anything within them (i.e. <p>) within the content column. We then combined content and title columns to receive a single column filled with the full text of each post, and lastly, we cleaned the text through tokenizing, lemmatizing, and removing stopwords.

After cleaning the data to receive a full data frame containing columns for every tag, we took the frequency counts of the tags in total. Since the number of tags is so vast, we simplified the number of tags to include the top 5% of tags regarding frequency within each data set. Once doing this, we were able to subset the final data frame to include only posts that contained at least one tag from the tags that categorized under the top 5% frequency-wise.

After manipulating and cleaning the text of the post, we created columns for every top 5% tag present within each dataset, all of which took values of either 0, which denoted tag absence within a post, or 1, which denoted tag presence.

Data Overview: Final Subsetted Frames per Category



## Algorithms & Explanation

### i. Gaussian Naive Bayes

Gaussian Naive Bayes is a type of supervised classifier typically used on features with continuous values. In addition to this, the classifier assumes normal distribution. Gaussian Naive Bayes uses the Bayes' theorem under the “naive” assumption of conditional independence between every pair of features given the value of the class variable. All in all, it predicts a class or label by estimating the probability of input values being associated with that class or label.

$$\text{Bayes' Theorem}$$

$$P(y | x_1, \dots, x_n) = (P(y)P(x_1, \dots, x_n | y)) / P(x_1, \dots, x_n)$$

### ii. Bernoulli Naive Bayes

Bernoulli Naive Bayes is a supervised classifying algorithm, which uses discrete binary/boolean data. Like Gaussian Naive Bayes, it uses Bayes' theorem under the "naive" assumption of conditional independence between every pair of features given the value of the class variable.

### *iii. Decision Tree*

A decision tree classifier is a supervised non-parametric classifier that splits the data into subsets by "decision nodes". These decision nodes dictate, which "leaf node" or classification route a value takes. Once going through each of the decision nodes, the value is lead to a leaf node, which serves as the prediction of the value.

### *iv. Extra Tree*

The extra tree classifier is similar to the decision tree classifier; however, it fits a number of randomized decision trees on various samples of the data frame.

### *v. Logistic Regression*

The logistic regression classifier uses the logistic function, or sigmoid function, to create models that fit its data.

$$\begin{array}{|c|} \hline \text{Sigmoid Function} \\ S(x) = 1/(1+e^{-x}) \\ \hline \end{array}$$

### *vi. K-Nearest Neighbors*

The k-nearest neighbors classifier outputs class. When a value is classified, it is classified by the most common classification among its k-nearest neighbors. For example, if  $k = 3$ , and the value's 3-nearest neighbors are classified as "Biology", "Biology", and "Cells" respectively, the value will be predicted as "Biology".

### *vii. Bagging*

A bagging classifier fits classifiers on random subsets of the data frame inputted with replacement and combine the predictions to form final predictions.

### *viii. Term Frequency-Inverse Document Frequency Matrix*

A term frequency-inverse document frequency matrix is a matrix in which rows are represented by m number of documents and columns are represented by n number of words within all of the documents. Like a document term matrix, the TF-IDF's elements are the frequency of each word within all individual documents; however, the TF-IDF calculates frequency while taking into account word-rarity. For instance, a specific word that shows up twice in all documents would have a higher frequency value than a word that shows up 10 times in all documents.

### *ix. Deep Learning via Neural Networks*

Neural networks uses several layers of processing to recognize patterns in data. The hidden layers of the neural network transform the original data into a model that we can use for classification. The connections between layers are weighted, and higher weights have more influence on other layers. We train the neural network with a large amount of data and then it can be used to classify features of new data.

## *Methods & Explanations*

### *i. Binary Relevance*

The binary relevance method decomposes a multi-label classification problem into  $n$  independent binary classification problems where  $n$  represents the number of labels. Scikit-multilearn's binary relevance function requires a base classifier for which Gaussian Naive Bayes will be used. Overall, different classifiers stated above will be used on each  $n$  independent binary classification problem, which binary relevance splits it into.

### *ii. Label Power Set*

The label power set method turns a multi-label classification problem into a multi-class problem. The label power set utilizes base classifiers by training it on all unique label combinations in the training data set.

## **Implementation Details**

### *Algorithm Implementation*

For both methods, power set and binary relevance, a random sample was taken of each (3,000 and 1,000 respectively) to accommodate for heavy computation times. For both, samples were taken from each of the category data frames and the combined data frame. From these samples of each of the data frames, we randomly took 70% of it to use as the training set and the remaining 30% to use as the testing set.

After splitting, the training data set's text was then used to create a term frequency-inverse document frequency matrix, which would be later used for algorithm implementation explained in the next section.

### *i. Multiple Classifiers via Power Set*

To implement Power Set, a classifier is set by initializing the function `LabelPowerset` through an appropriate classifier. Next, the classifier was trained and fit through the training set, which was then used to create predictions from the testing set via trained classifier. Lastly, F-score was calculated by comparing predictions versus actual observations and through weighted average. [Applicable Classifiers: Logistic Regression, Extra Tree, Bagging, Bernoulli Naive Bayes, K-Nearest Neighbors, Decision Tree]

### *ii. Multiple Classifiers via Binary Relevance*

Implementation of Binary Relevance exactly mirrored that of Power Set; however, the classifier was set by initializing the function `BinaryRelevance` through an appropriate classifier. [Applicable Classifiers: Gaussian Naive Bayes, Extra Tree, Decision Tree, Bagging, Bernoulli Naive Bayes, K-Nearest Neighbors]


### *iii. Neural Networks*

Furthermore, we used an open source neural network library called Keras for classification using convolutional neural networks. Only the top 100 tags were considered. In our model's loss function, we factored in class weights for each tag class to address tag imbalance. Word embeddings represent each token as a low dimensional vector; and we obtained word embeddings from a neural network. We used a 1D convolutional neural network with a convolutional layer of window size 3. The neural network model learns word sequences in text, and it can be used to recognize these word sequences in other texts. We input the post text and title into the model and the output was a dictionary of all tags and its corresponding probability. The probability represents how likely the post has that tag. The tag was assigned to the post if the tag has a probability of 30% or more. If no tags satisfied this criteria, we only assigned a single tag with the highest probability. Additionally, we limited each post to have maximum five tags. After predicting the tags for each post, the F-score was calculated by comparing one hot encoding matrices of the original tags and the predicted tags.

Computation Warning: Depending on method, classifier, and data frame, computation time can range from nearly instant to 3+ hours.

## Results

### Concrete

 : Best Performing Classifier by Column

 : Best Algorithm per Table

 : Gets Timed Out

#### Power Set Method

<i>F-Score Table</i>	<u>Biology</u>	<u>Cooking</u>	<u>Crypto</u>	<u>D.I.Y.</u>	<u>Robotics</u>	<u>Travel</u>	<u>Full</u>
<b>Logistic Regression</b>	0.0883	0.0587	0.1292	0.0547	0.0876	0.0553	0.0129
<b>Extra Tree</b>	0.0879	0.0997	0.1336	0.106	0.1215	0.078	0.0503
<b>Bagging</b>	0.1299	0.1629	0.1891	0.1551	0.2077	0.1172	0.0761
<b>Bernoulli Naive Bayes</b>	0.0281	0.0105	0.0338	0.0373	0.0242	0.0278	0.0024
<b>K-Nearest Neighbors</b>	0.1968	0.2731	0.2615	0.2665	0.2333	0.2428	0.1893
<b>Decision Tree</b>	0.1071	0.154	0.1893	0.1443	0.184	0.0991	0.0646

#### Binary Relevance Method

<i>F-Score Table</i>	<u>Biology</u>	<u>Cooking</u>	<u>Crypto</u>	<u>D.I.Y.</u>	<u>Robotics</u>	<u>Travel</u>	<u>Full</u>
<b>Gaussian Naive Bayes</b>	0.0737	0.0481	0.079	0.1022	0.12	0.065	0.0319
<b>Extra Tree</b>	0.0880	0.0923	0.1186	0.1102	0.1982	0.1055	0.0455
<b>Bagging</b>	0.1278	0.265	0.2501	0.1603	0.2898	0.1927	0.1047
<b>Bernoulli Naive Bayes</b>	0.0427	0.045	0.0748	0.0753	0.0923	0.0418	0.0058
<b>K-Nearest Neighbors</b>	0.1586	0.294	0.2673	0.2671	0.2829		
<b>Decision Tree</b>	0.1869	0.2907	0.2951	0.243	0.3177	0.2297	0.1551

Best Algorithms per Method

<i>F-Score Table</i>	<u>Biology</u>	<u>Cooking</u>	<u>Crypto</u>	<u>D.I.Y.</u>	<u>Robotics</u>	<u>Travel</u>	<u>Full</u>
<b>Binary Relevance -</b> Decision Tree	0.1869	0.2907	0.2951	0.243	0.3177	0.2297	0.1551
<b>Power Set -</b> K-Nearest Neighbors	0.1968	0.2731	0.2615	0.2665	0.2333	0.2428	0.1893
<b>Deep Learning -</b> Neural Networks	0.3544	0.5065	0.428	0.5286	0.2204	0.5682	0.4915

**Results Summary:** Using neural networks, the best overall algorithm, our various categories' F-scores ranged (**0.2204 ~ 0.5286**). Regarding the full frame, our F-score was **0.4915**.

### *Lessons Learned & Insights*

To summarize, from this project we've familiarize ourselves with two different methods, binary relevance and label power set, to deal with a multi-label classification problem. In addition to this, we've learned about and how to implement multiple base classifiers such as logistic regression, Gaussian Naive Bayes, Bernoulli Naive Bayes, extra tree, decision tree, bagging, and K-nearest neighbors voting to train a classifying model to later create classification predictions.

Additionally, we attempted to use transfer learning on a Google transfer learning data set via a natural language processing technique called Bidirectional Encoder Representations from Transformers (BERT), but we were unable to use it successfully and obtain results. Transfer learning is the use of a pre-trained machine learning model on new datasets without training from scratch. We have included the code for reference.

Furthermore, we have learned how to use neural networks for text classification. Neural networks outperformed almost all other types of classification algorithms for every category of posts. Thus, in the context of text classification, it seems that deep learning algorithms, like neural networks, have better accuracy when compared to other methods. However, a disadvantage to neural networks is that it is difficult to understand why it generates predictions because of its black box nature.

Overall, for this multi-label classification problem, neural networks performed the best in classifying tags to each post. As shown in the "Best Algorithms per Method" table, although neural networks worked best overall, binary relevance via decision trees beat neural networks in the category "robotics". Therefore, this shows that there isn't one "best" method or algorithm, and the performance of classification depends on the situation or different factors of a dataset.

### *Limitations*

- The binary relevance method is limited in that it does not take into account relationships between labels. For example, if a label shows up, it does not take into account the likelihood a related label will show up as well.
- Even after cutting tags down to include only the 95<sup>th</sup> percentile, there were still so many tags to classify that it took more of a toll on the F-scores of our bigger data sets (such as travel or full)
- In the event that a tag in the testing set is not within the training set, this tag won't accurately be able to be predicted since it won't be trained in the model
- Due to the large computation power that the binary relevance method and K-nearest neighbors algorithm has on the biggest data frames, F-scores of the travel and full data frames were not able to be computed for this method and algorithm combination.
- If possible, we could further our attempts of getting transfer learning via BERT to work successfully.
- The data contains a substantial number of features, and we may want to further refine our feature selection methods to reduce computations.

## References

### [General]

- <https://www.kaggle.com/c/transfer-learning-on-stack-exchange-tags/data>
- <https://towardsdatascience.com/journey-to-the-center-of-multi-label-classification-384c40229bff>

### [Methods]

- Binary Relevance
  - [http://scikit.ml/api/skmultilearn.problem\\_transform.br.html](http://scikit.ml/api/skmultilearn.problem_transform.br.html)
- Label Power Set
  - [http://scikit.ml/api/skmultilearn.problem\\_transform.lp.html](http://scikit.ml/api/skmultilearn.problem_transform.lp.html)

### [Algorithms]

- Bagging
  - <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html>
- Bernoulli Naive Bayes
  - [https://scikit-learn.org/stable/modules/generated/sklearn.naive\\_bayes.BernoulliNB.html](https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html)
- Decision Trees
  - <https://medium.com/@chiragsehra42/decision-trees-explained-easily-28f23241248>
- Extra Tree
  - <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html>
- Gaussian Naive Bayes
  - <http://dataaspirant.com/2017/02/20/gaussian-naive-bayes-classifier-implementation-python/>
- K-Nearest Neighbors
  - [https://en.wikipedia.org/wiki/K-nearest\\_neighbors\\_algorithm](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm)
- Logistic Regression
  - [https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.LogisticRegression.html](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)
- Naive Bayes
  - [https://scikit-learn.org/stable/modules/naive\\_bayes.html](https://scikit-learn.org/stable/modules/naive_bayes.html)
- Neural Network
  - <https://blog.mimacom.com/text-classification/>
- Transfer Learning
  - <https://towardsdatascience.com/building-a-multi-label-text-classifier-using-bert-and-tensorflow-fl-88e0ecdc5d>