**Automating Retrieval and Classification of News Articles**

**Gautam Visveswaran, Evgenia Kitaevich**

**Abstract**

The goal of this project was to automate the process of building a dataset of newspaper articles relevant to the Armenian government’s treatment of its citizens during its period of conflict. In order to accomplish this, newspaper articles were scraped from various sources and were classified using a support vector machine (SVM) model. The classifier was judged based on how accurately it determined the relevance of newspaper articles, and by how accurately it classified each article as one of several predetermined categories. Our final result was a classifier which predicted the relevance of articles with a raw accuracy of 85.1% and had an F1 score of .418. The training dataset of articles was not large enough to draw a meaningful conclusion about the classifier’s performance in categorizing articles, since several categories were underrepresented.

**Introduction**

The nation of Armenia has been in conflict with its neighbor, Azerbaijan, for decades. The issue has largely been due to disputes over the Nagorno-Karabakh region, which the two nations went to war over in 1988. Despite a ceasefire in 1994, there have been poor relations between the two, and there is often violence that occurs along the border. As a result, the citizens of Armenia have had to deal with living in wartime conditions. One of the primary goals of government is to provide aid and services to its citizens, especially during such a time. Our team focused on the provision of public goods in particular. Determining scenarios in which the government is not able to provide public goods for its citizens allows researchers to propose solutions or alternative approaches to ensure the public wellbeing.

Our team used newspaper articles as a primary source of determining the effectiveness of the government. As the conflict has spanned multiple decades, one objective of the group was to compile a large dataset of articles pertinent to the conflict, in order to be studied. However, the process of manually searching for and reading through articles to extract meaningful information is very time consuming in nature. My aim during the course of this project was to automate the process of expanding the dataset. Specifically, the tasks undertaken were to:

* Scrape websites that contained Armenian news articles
* Process the articles retrieved
* Classify the articles as relevant or irrelevant to the conflict
* Categorize the articles

Building such a program would reduce the time taken to find articles and would filter out several useless articles, such that the research team was left with a concise, relevant dataset of labeled articles to work with.

**Methods:**

The first step in the process was to decide how to classify the articles. In this case, I decided to use an SVM and a bag-of-words approach. An SVM is a machine learning tool used to classify information. To train the model, the machine takes an NxD matrix as input, in which each row is a vector of features and each column represents a unique feature. In addition, the SVM takes in a 1xN vector consisting of the correct labels for each of the input feature vectors. With this information, the model builds a boundary in an N-dimensional space. Once the model is trained, it can take in any NxD matrix of input features and predict the classification of each input vector. Shown below is a visual representation of the SVM boundary with 2 dimensions.

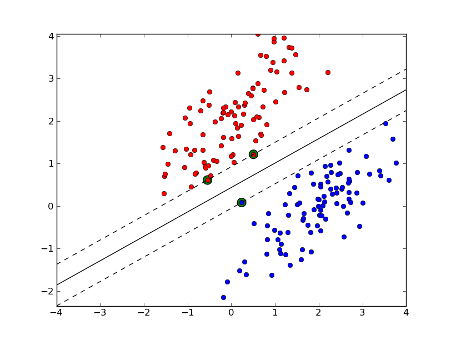


Figure 1: Diagram of SVM boundary

The bag-of-words model is a representation used in natural language processing in which the frequency of each word in a text corpus represents a unique feature. Therefore, in this case, each news article represented a row in the SVM, and each column represented a word that was present in the corpus of articles.

Given this choice of model, I chose to use Python (version 2.7.15) to write the project in, as there were several existing libraries that were very relevant to this task. The process was broken down into three distinct steps, scraping, processing, and classifying, all of which are described below.

**Scraping:**

The datasets used for this report are newspaper articles from panarmenian.net, rferl.org, and armenpress.am. Scrapers were written for both RFERL and Armenpress, while the articles from Panarmenian had in large part already been gathered by the team. Scraping was accomplished using Selenium, a tool which is used to automate web browsers. The software is primarily used for testing, but in this case, was very helpful in scraping when it was necessary to interact with JavaScript based elements on the websites. Once a list of URLs with the articles was compiled, the source HTML for each of the URLs was retrieved and parsed using BeautifulSoup, a Python HTML parser. Information stored for each article included the title and text of the article, as well as the date, location, and other metadata for the team to work with. This content was stored in a series of CSV files which were then used in the next steps.

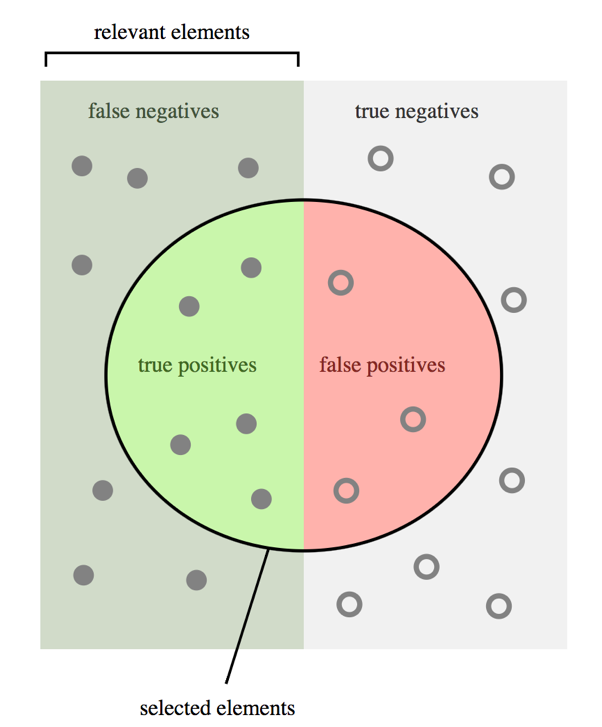
**Processing**

Once all text from the articles had been retrieved, the next step was to preprocess the text such that only relevant words were used as features in the SVM. Stopwords, or words that were likely to be found in every article and would therefore not alter the performance of the classifier, were removed. Additionally, a stemming library (specifically the PorterStemmer) was used in order to extract the roots of words in the articles. This was done to ensure that words with the same root but different endings would not be counted as separate features. Finally, punctuation and other characters were removed and a dictionary object was created from the resulting list of word tokens to be used with the classifier.

**Classifying**

The classification step involved creating the feature vectors from the articles, which was accomplished by preprocessing each article and iterating through each word, counting the number of occurrences of that word. Vectors were created using the NumPy library and were then fed into the SVM. The Scikit-Learn library’s implementation of the SVM algorithm was used. Two classifiers were written, one to predict the relevance of the articles and another to predict which category the article belonged to. These categories included “Public Good Provision”, “Private Good Provision”, “Army/Security”, “Nagorno-Karabakh Region Installments”, “External Economic Events”, “International Aid”, “Protests”, “Political Deaths/Attacks”, and finally a “Junk” category. Both classifiers took the same feature vector as input but were trained with different label matrices.

**Measuring Accuracy**

The performance of the classifier was judged in two distinct ways: by finding the raw accuracy of the classifier (correct predictions / total predictions) and, in the case of the binary classification of relevance, by finding the F1 score. The F1 score is a metric used to determine accuracy of a binary model and is calculated by taking the harmonic mean of the precision and recall of the model. Precision is defined as the total number of true positives divided by the sum of the true and false positives (TP / TP + FP). Recall is defined as the number of true positives divided by the sum of the true positives and false negatives (TP / TP + FN). F1 score is calculated using the formula F1 = (2 \* Precision \* Recall) / (Precision + Recall). This comprehensive score lies in a range between 0 and 1, with 0 representing a completely incorrect classifier and 1 representing a perfect classifier. The measure is generally more useful in gauging classifier performance, especially when there is an uneven class distribution. Figure 2 depicts precision and recall.

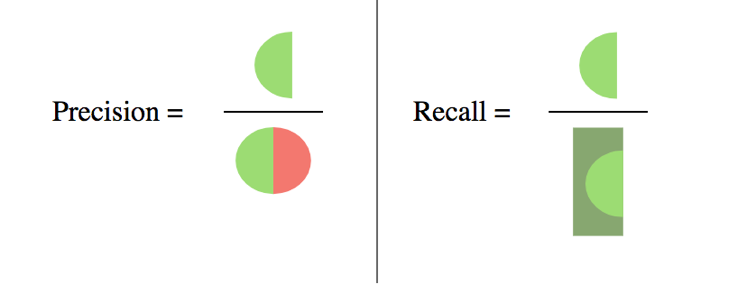


Figure 2: Visualization of Precision and Recall

K-folding was used in order to determine the accuracy of the training data. This method is used to predict the performance of the classifier on future testing data. The original data is split into a training set and a testing set; the classifier is trained on the training set, while the test set is used to determine how the classifier performed. This process is repeated k times, with the mean performance reported as the final accuracy. In this case, k was set to 5, and during each iteration, 80% of the training data was selected to train with and 20% was used to test with. K-folding is illustrated in Figure 3.

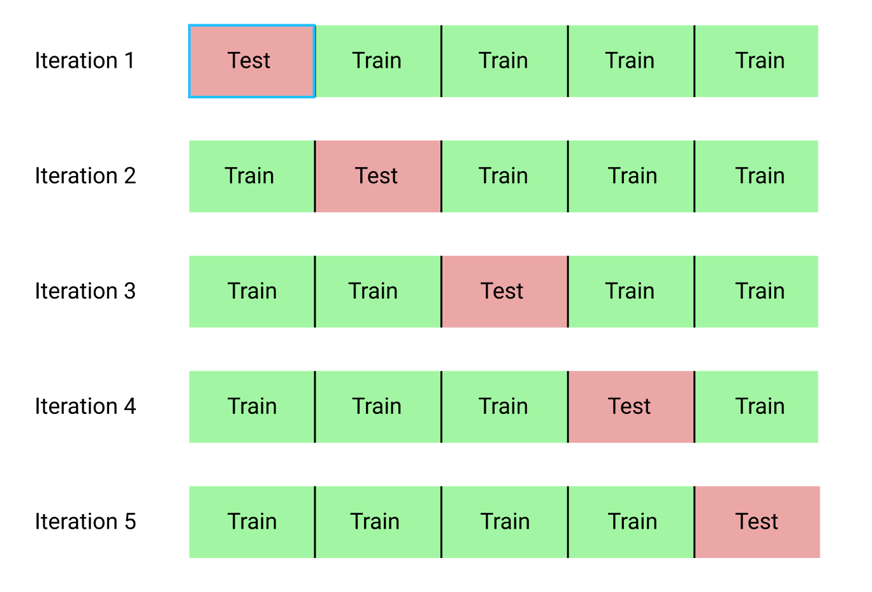


Figure 3: The K-Folding Process

**Data and Results**

During the initial run of the classifier, the dataset consisted of only 156 articles. In addition, only 30 articles were relevant and of these articles, there were very few per category. As a result, the classifier simply predicted that every article was irrelevant and belonged to the “Junk” category. This led to a raw accuracy score of 80.8%, but the F1 score for relevance was undefined, as no positive examples were predicted by the SVM. This result clearly showed that the training dataset was far too small in size, rendering the classifier useless. The solution to this issue was to expand the size of the training dataset by reading and manually classifying more news articles. The team dedicated a significant amount of time doing so, and for the second pass, had comprised a dataset consisting of 958 articles.

The SVM’s C-value hyperparameter was also tuned in order to improve performance. Hyperparameters are configurable inputs which are set by the user prior to training the model. The C-value represents the extent to which misclassified examples are penalized, and therefore changing the C-value changes the margin boundary of the SVM, leading to some points being classified differently. A low C-value dictates a larger minimum distance between correctly classified points and the boundary. A high value represents a smaller distance but increases the number of correctly classified points in the learning set. Figure 4 depicts the difference between a high and low C-value. Several different C-values were tested until the value that led to the best accuracy and F1 score was found.

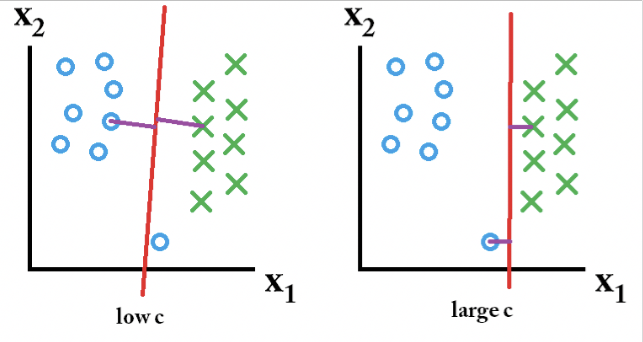


Figure 4: C-Value Visualization

Given the larger training set and newly tuned hyperparameter, the model now made much more realistic predictions, producing an accuracy of 85.1% and an F1 score of .418 when determining whether an article was relevant or not. The classifier for categorizing articles, however, was still not particularly useful, since certain categories were still not very highly represented in the training set.

In addition to finding the accuracies, the program also output the words that had the highest weight in the SVM and were therefore most likely to be present in relevant articles. These words, included “Yerevan”, “police”, “line”, “contact”, “Gyumri”, “protest”, “Azerbaijani”, “demonstrate”, “energy”, “rally”, “end”, and “ministry”.

**Future Steps**

The biggest issue faced during the course of the project was the small size of the training dataset. While there was enough data by the end to have a reasonably well-performing classifier for relevance, there was not enough data to accurately categorize the articles. Immediate next steps would be to continue growing the training dataset, with a focus on adding more articles that involve some of the less common categories. As the size of the training dataset increases, the performance of this particular model will increase as it approaches an asymptote.

**References**

Azab, Mahmoud, et al. “Analysing RateMy-Professors Evaluations Across Institutions, Disciplines, and Cultures: The Tell-Tale Signs of a Good Professor.” SpringerLink, Springer, Cham, 11 Nov. 2016, link.springer.com/chapter/10.1007/978-3-319-47880-7\_27

Mirovalev, Mansur. “Here’s Why a ‘frozen’ conflict between Armenia and Azerbaijan has gotten hot”. LA Times, 16 Apr. 2016. <http://www.latimes.com/world/europe/la-fg-nagorno-karabakh-20160419-story.html>

Porter, Martin. “The Porter Stemming Algorithm” Jan. 2006. <https://tartarus.org/martin/PorterStemmer/>

“Predicting Yelp Stars from Reviews with Scikit-Learn and Python.” DevelopIntelligence Blog, 8 Apr. 2017, [www.developintelligence.com/blog/2017/03/predicting-yelp-star-ratings-review-text-python/](http://www.developintelligence.com/blog/2017/03/predicting-yelp-star-ratings-review-text-python/)

**Image Sources**

<https://i.stack.imgur.com/QCFUZ.png>

<https://stats.stackexchange.com/questions/31066/what-is-the-influence-of-c-in-svms-with-linear-kernel>

<https://upload.wikimedia.org/wikipedia/commons/2/26/Precisionrecall.svg>

<https://my.oschina.net/Bettyty/blog/751627>