**Report: Topic Modelling and Question Answering in Natural Language Processing Using Machine Learning Algorithms**

Ekiyor Odoko, Alex Waweru, Emmanuel Nimo, Daniel Olukoya, Ernest ThankGod  
Department of Computer Science, Department of Computer Science, Department of Computer Engineering, Department of Computer Science, Department of Computer Science  
[ekiyor.odoko@ashesi.edu.h](mailto:ekiyor.odoko@ashesi.edu.h), | [alex.waweru@ashesi.edu.gh](mailto:alex.waweru@ashesi.edu.gh), | [emmanuel.nimo@ashesi.edu.gh](mailto:emmanuel.nimo@ashesi.edu.gh), [daniel.olukoya@ashesi.edu.gh](mailto:daniel.olukoya@ashesi.edu.gh), | [ernest.thankgod@ashesi.edu.gh](mailto:ernest.thankgod@ashesi.edu.gh)

**Abstract**

This document details the processes taken to create a program that predicts the category of input documents and predict answers to questions. For each task in this project, we come up with 3 different ideas and implement them.

**Introduction**

Topic modelling refers to the text-categorization task of determining what topic or category does a question fall under. Question Answering is simply referring to the task of providing an answer to set of input questions.

**Methodology**

For this project, we are to come up with 3 different ideas for each task, discuss them, implement them, and finalize the best solution. For the topic modelling tasks, our ideas are;

* **K Nearest Neighbor** – We used a KNN classifier for multinomial classification. The classifier utilizes a cosine similarity function to check for the closest K similar questions. The topics of the questions are retrieved, and the mode topic is returned.
* **Logistic regression** - We used a logistic regression classifier for multinomial classification. This was partly because w had some trouble implementing neural networks. Such a classifier would work by doing multiple binomial classification for each class with the class as one class and all the other classes as another class. After this it returns the class with the highest probability for the given question. The questions and answers were therefore pre-processed, and all numbering and unnecessary whitespace removed. A count vectorizer was then used to extract a vector of features from the questions which had all been made lowercase in the program. The extracted vectors of features and their labels were then used to create a logistic regression model to classify question inputs.
* **Naïve Bayes –** Initially, we convert the questions to a vector of counts and then use scikit-learn’s library to transform the count vectorized format to a term frequency-inverse document frequency form. We then pass the new vector as features to a multinomial naïve bayes classifier that creates likelihoods for each feature as well as the priors. We use the trained model to now make topic predictions for each input question.

For the question answering task, the three ideas implemented are;

* **Cosine Similarity** - In this method, both documents (questions.txt, answers.txt) were represented as vector of tf-idf weights to extract features from the datasets, and the cosine angles between the document vectors were calculated to relate them. Each one of the documents had about 2608 separate statements. A command line argument is incorporated to pick up questions and the program returns the most similar answer to the question. For instance, in the example below, the question “Who is the head of state of Ghana?” is matched with the most similar answer “Nana Akuffo Addo”.
* **Jaccard Weighted Similarity** – In this method, a vector representation of the question is calculated. For each word in the test document, the index of the word in the unique words set from the test documents is used as a representation of the presence of the word. The method finds jaccard similarity between the test sentence and all the documents in test case. It retrieves the answer to the most similar question/document

**Jaccard(a,b) = intersection(a,b)/union(a,b)**

* **Minimum Edit Distance** – In a typical minimum edit distance situation, you’re calculating the minimum changes in characters between a source text and target text, but in this implementation, we look at it at a word level. For every question, we compare it against the questions in our training set. And then calculate the minimum edit distance for each word positions in the both questions.