Classification of Ailments Given Description of Symptoms

Urmzd Mukhammadnaim
Ben MacDonald

December 6, 2021

Contents

Abstract

In this paper, we address the challenges experienced in the preliminary research phase of ailment diagnosis performed by many individuals prior to visiting a healthcare professional. Due to the large quantity of varying results appearing once someone searches their current symptoms, we developed a Convolutional Neural Network (CNN) that reduces this clutter by returning only the most probable medical condition given the user's description. Forthcoming, we decided to limit the possible classifications to the medical conditions, migraine, tetanus and depression. To train the model, data is obtained by scraping text data from various medical websites which were written using casual terminology to form a corpus. The corpus was then vectorized using one-hot encoding in one implementation and the FastText model in another. Once the model is trained, we then analyzed its ability to classify our own descriptions of migraines, depression and tetanus.

Chapter 1

Methodology

1.1 Data Processing

After the extraction and consequent concatenation of the documents for each classification, we explored two preprocessor implementations. We first describe the use of One Hot Encoder and Witten-Bell Probability Distribution to retain morphological-level information, and generate unique sets of words that can later be fed into the CNN. Subsequently, we analyze the use of the FastText model as a means of retaining semantic-level information, and ensuring words with similar words appear closer together in the subspace.

1.1.1 One Hot Encoding and Witten Bell Generation

After the documents were tokenized, the text was piped into the transformer $T1$ which applied a series of filters and maps to remove punctuation and ultimately break the words into their stems. The result of $T1$ consisted of a set of unique stems D_n where n represents the order of which the the document was processed in. After $T1$ was applied to all documents, the union of all the sets was taken to form the vocabulary V. V was then fed into the *FreqDist* class from the *nltk.probability* package to allow the subsequent usage of the *WittenBellProbDist* class. By allocating $|V|$ bins, and using the *FreqDist* of D_n, we ended up with an instance of the *WittenBellProbDist* capable of generating ${}_VC_S$ unique sets, where S represents the cardinality of the generated set. The described method was used to prevent clustering of the same words, and ultimately prevent the model from developing an aptitude for classifing an input based on the frequency of its instances instead of the existence of an instance.

After N samples were generated via the *WittenBellProbDist* method described above, we transformed the data using an instance of the *sklearn.preprocessor.OneHotenco*
class fit on the the vocabulary V. With this each word in D had its own unique binary vector and out-of-vocabulary (OOV) words were treated as the 0 vector with size $|V|$.

1.1.2 FastText Model

In the alternative preprocessor, we used the FastText model. Unlike the former method, no data generation was done. Instead, all documents were first tokenized by sentence using an instance of the *nltk.tokenizer.PunktSentenceTokenizer*. Afterwards, an implementation of the FastText into an implementation of the