**Experiments with features**

**TFIDF**

* Try different n gram’s

**Word2Vec**

* Use Google pre-trained model and create features based on that for the current dataset
* Fine-tune the google pre-trained model for our dataset: Problem of using pre-trained word embedding is that they are unable to deal with out-of-vocabulary (OOV) words, i.e. words that have not been seen during training. Typically, such words are set to the UNK token and are assigned the same vector, which is an ineffective choice if the number of OOV words is large.
* Training from scratch (Won't really work because the neural network needs a lot of data to learn the weights)
* For parameters we can fine-tune the model based on both Continuous bag of words and skip grams.

**Experiment with models (The following should be tested with all the possible features experimented above**

**SVM**

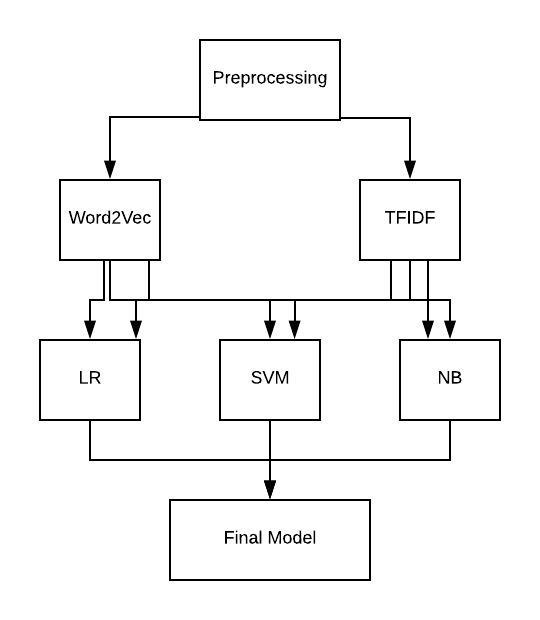
* Different kernel functions

**Logistic Regression**

* Hyper parameters (L1,L2)

**Naïve Bayes**

**Model Pipeline**



**Note: The experiments will be done for both article headlines and content separately.**

**Experiment 1 (Done):**

**Text to Features:** TFIDF using the article content (Try different n grams)

**Classification Model:** Logistic Regression (Different Penalties)

**Experiment 2 (Done):**

**Text to Features:** TFIDF using the article content (Try different n grams)

**Classification Model:** Support Vector Machine

**Experiment 3: (Done)**

**Text to Features:** TFIDF using the headlines

**Classification Model:** Logistic Regression (Different Penalties)

**Experiment 4: (Done)**

**Text to Features:** TFIDF using the headlines

**Classification Model:** Support Vector Machine

**Experiment 5:**

**Text to Features:** Word2Vec (Hyper parameter)

**Classification Model:** Support Vector Machine

**Experiment 6:**

**Text to Features:** TFIDF, Keywords, Length of each article

**Classification Model:**

**Other techniques that might help:**

* Imbalanced classes
* Different preprocessing

**Parameters to check**

Since our goal is to recognize fake news articles, the ones we correctly classified as fake are our True Positives, and the fake articles we incorrectly classified as real are our False Negatives (**Type II error**). Real articles that we correctly classified are our True Negatives and incorrectly classified real articles are our False Positives (**Type I error**).Our goal is to minimize both the False Negatives and False Positives. The F1 score helps strike a balance between precision (fake articles classified correctly over the total number of articles predicted as fake) and sensitivity/recall (the proportion of fake articles classified correctly). For that reason, we will use the F1 metric as our optimization parameter when using cross-validation to tune our hyper parameters.