Capstone Project 2: Milestone Report
Spring Board Data Science Career Track
~ By Zach Nguyen

Data Story

Business Problem: Nowadays, companies juggle with quite a few product brands. Despite their maintenance cost, product brands are strong value generation engines for many established companies. For this reason, companies are constantly looking out across Blogs, Forums, and other Social media platforms, etc. to check the sentiment for their various products and competitor products to learn how their brand resonates in the market. This type of Sentiment analysis helps them as part of their post-launch market research to determine the effectiveness of the brand of the product they have released into the market.

Client: The client is a pharmaceutical company who is interested in understanding how their drug products are doing. Even though this problem is significant in the medical industry, it can be very similar in many other consumer-driven industries like food, clothing retail, e-commerce etc .. With a good algorithm, companies can flag good and bad products for review and generate real-time metrics on how their products are useful or non-useful to consumers.

In the scenario played by this report, I will assume the role of a data scientist of a pharmaceutical company to conduct and assess the use of various sentiment analysis techniques on the collected dataset. I will then attempt to bridge the results of the Data Science/ Machine Learning pipeline with the realistic requirements of the company in order to make business recommendations on how to proceed. The guiding questions of the project are as followed:

- Is the collected data useful now for the task of sentiment discrimination? Is it enough? Will it be in the future?
- Can the data be valuable elsewhere?
- Should the company invest in collecting more data? Should it invest in better infratructures for Machine Learning?

Data source: The data found is from a sentiment analysis[1] hackathon hosted by Analytics Vidhya. It was collected from a real pharmarceutical and is an excellent example of how messy and challenging text data can be, even for a very traditional task of sentiment analysis. The data contains reviews of over 100 different drugs with 3 sentiment labels (positive, negative and

[1] Retrieved From: https://datahack.analyticsvidhya.com/contest/innoplexus-online-hiring-hackathon/

neutral). Due to the of the confidentiality of the data and the fact that the competition is over, the data cannot be shared for reproduceability.

Methodology:

- Use text analysis techniques to explore the data and address any data preprocessing and data imbalance (with packages such as Pandas, Seaborn, nltk, gensim or native arguments in algorithm).
- Build and explore various modern Natural Language Processing (NLP) techniques to improve text classification (using scikitlearn for shallow algorithms and gensim, keras-tensorflow for neural-based techniques).
- Devise a business plan and recommendation for the use case of the algorithm.

Deliverables: Slide Deck, Report, Code

Data Wrangling and Preprocessing

After defining and using a simple function to check random reviews of different sentiments. One of such reviews can be seen here:

As can be seen, a few aspects were discovered that could hinder the quality of the data:

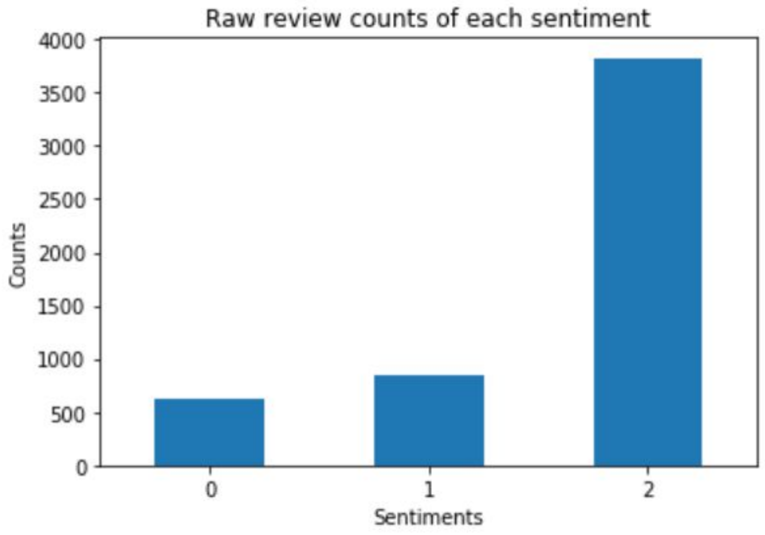
- None of the reviews are missing, so removal or imputation is unnecessary.
- The sentiments have three classes (0: positive, 1:negative, 2:neutral), thus multi-class algorithms are a must.
- The reviews include a high data imbalance, which could be rectified with resampling techniques to including appropriate weights in our algorithm.
- Many reviews and are confusing to classify, even for humans! This is due to the fact that a review could talk about other drugs (and therefore, could express different sentiment than the drug in question).
- The reviews include a lot of punctuations and numerics and occasional links/urls that could hinder Bag of Word analysis.
- The drug-name sometimes include hyphens and numerics, so caution must be taken to ensure the drug name remains in-tact.
- The reviews include a lot of contractions, which could create inaccurate lemma (example: "don't" is not the same as "do" when lemmatizing).

To resolve the above issues, the steps performed for data cleaning include:

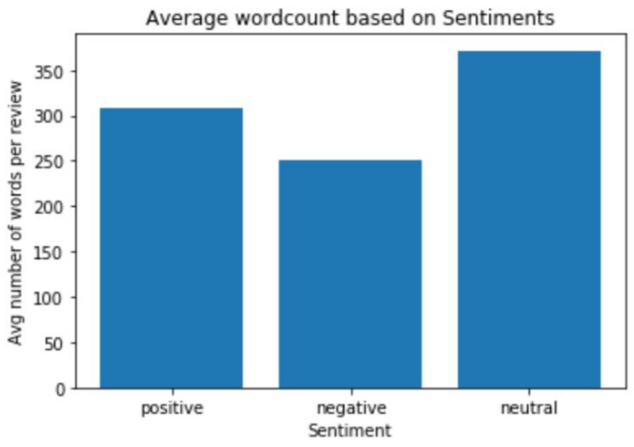
- Convert the text to lower-cased.
- Expand all contractions using a custom contraction mapping dictionary.
- Remove all links.
- Remove all punctuation except hyphens.
- Remove all numerics and hyphens which stand by themselves (between two spaces).
- Lemmatize each document with respect to its Part-of-speech using a lemmatizing function.

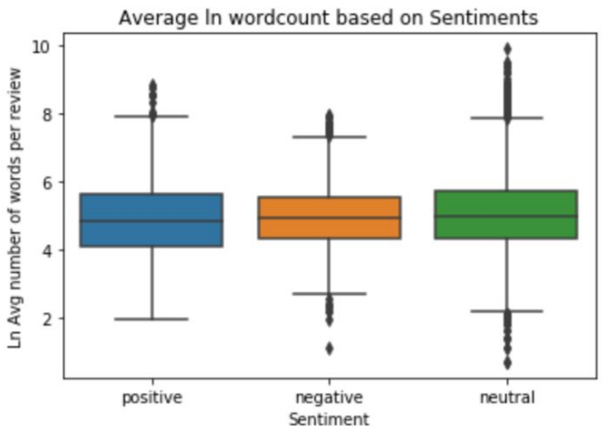
Exploratory Data Analysis

1. **Sentiment break-down:** Our sample displays high imbalance with a strong bias towards the neutral class. This tendency is surprising, since people don't usually write reviews for products they feel neutral about. Note, however, that the reviews here contain comments to other reviews as well. The reasonable next step is to look at the word count of sentiments.



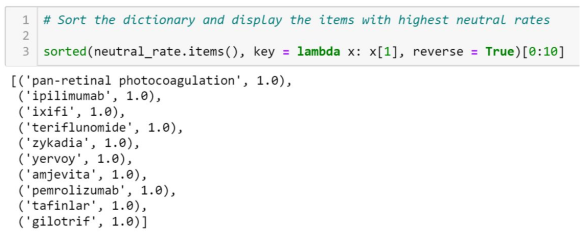
2. **Average Word Count:** There looks to be a difference in the average word count of the different sentiments. Specifically, negative reviews tend to be more curt and neutral reviews more verbose. However, upon closer inspection using boxplot, we can see that the median wordcount and the interquartile ranges is quite close among all three sentiments. This suggests that neutral reviews tend to be more on the extreme.





A question arises as to whether the average word count is really statistically significant. Using the Kruskal-Wallis H test (The ANOVA one way cannot be used due to its assumptions of normality, which is clearly violated with our samples), we can be confident that they are with F-stat of 11.9 and a p-value of less than 0.05.

3. **Innate drug quality:** A question arises to whether each drug has innate quality that allows sentiment on them to be judged as negative/positive/neutral. Investigation shows that there are drugs with correspondingly 100% positive/negative/neutral reviews. Thus, it can be concluded that what type of drugs being reviewed can affect the sentiments. Here is an example of 10 drugs which only has neutral reviews:

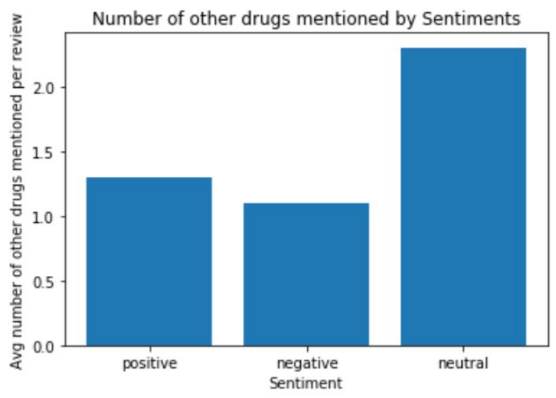


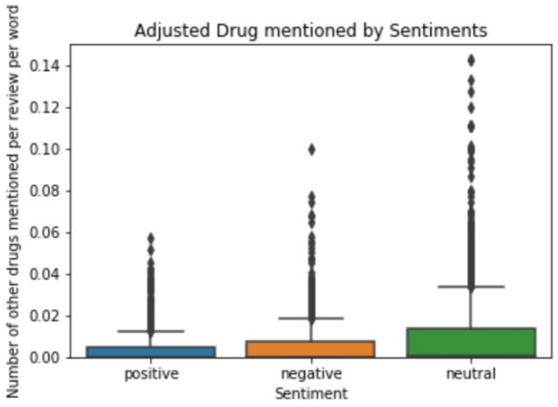
```python
# Sort the dictionary and display the items with highest neutral rates

sorted(neutral_rate.items(), key = lambda x: x[1], reverse = True)[0:10]
```

```
[('pan-retinal photocoagulation', 1.0),
 ('ipilimumab', 1.0),
 ('ixifi', 1.0),
 ('teriflunomide', 1.0),
 ('zykadia', 1.0),
 ('yervoy', 1.0),
 ('amjevita', 1.0),
 ('pemrolizumab', 1.0),
 ('tafinlar', 1.0),
 ('gilotrif', 1.0)]
```

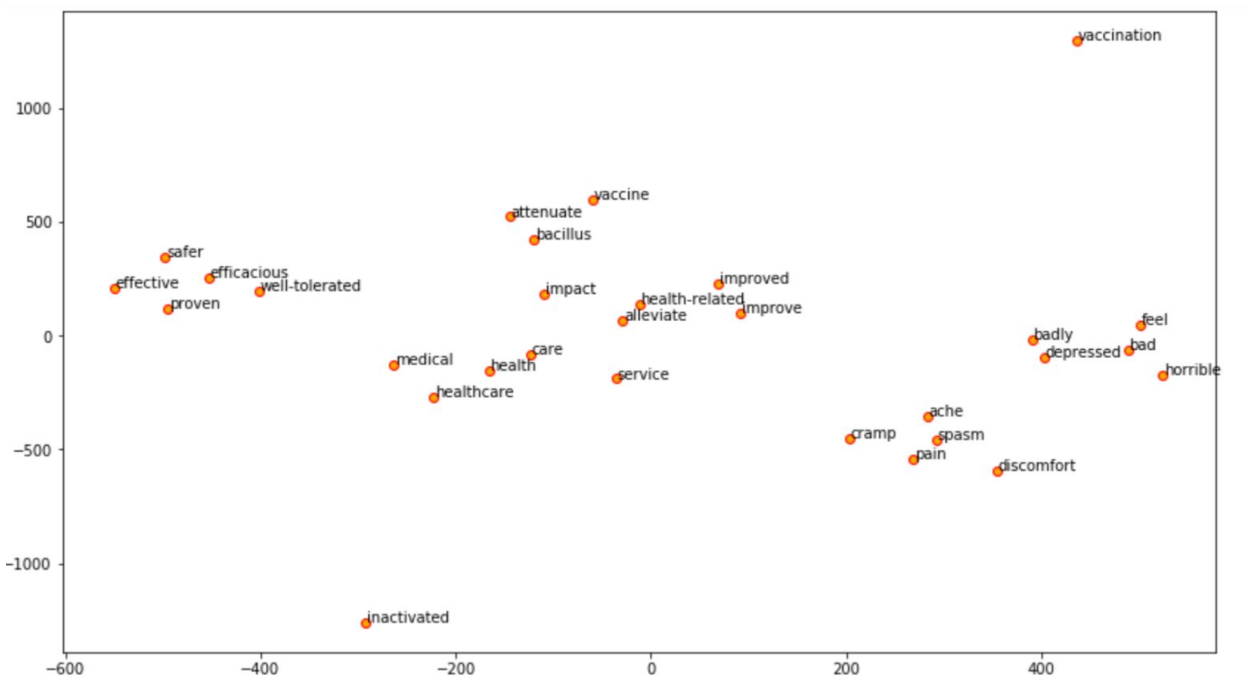
4. **Number of drugs mentioned adjusted for length of review:** Another question arises to whether or not other drugs mentioned in the review contribute to its sentiments. An example is a review comment which lists out many different types of drugs as an objective (neutral) suggestion. Neutral reviews seem to have high drug mentions, even when adjusted for number of words in such review.





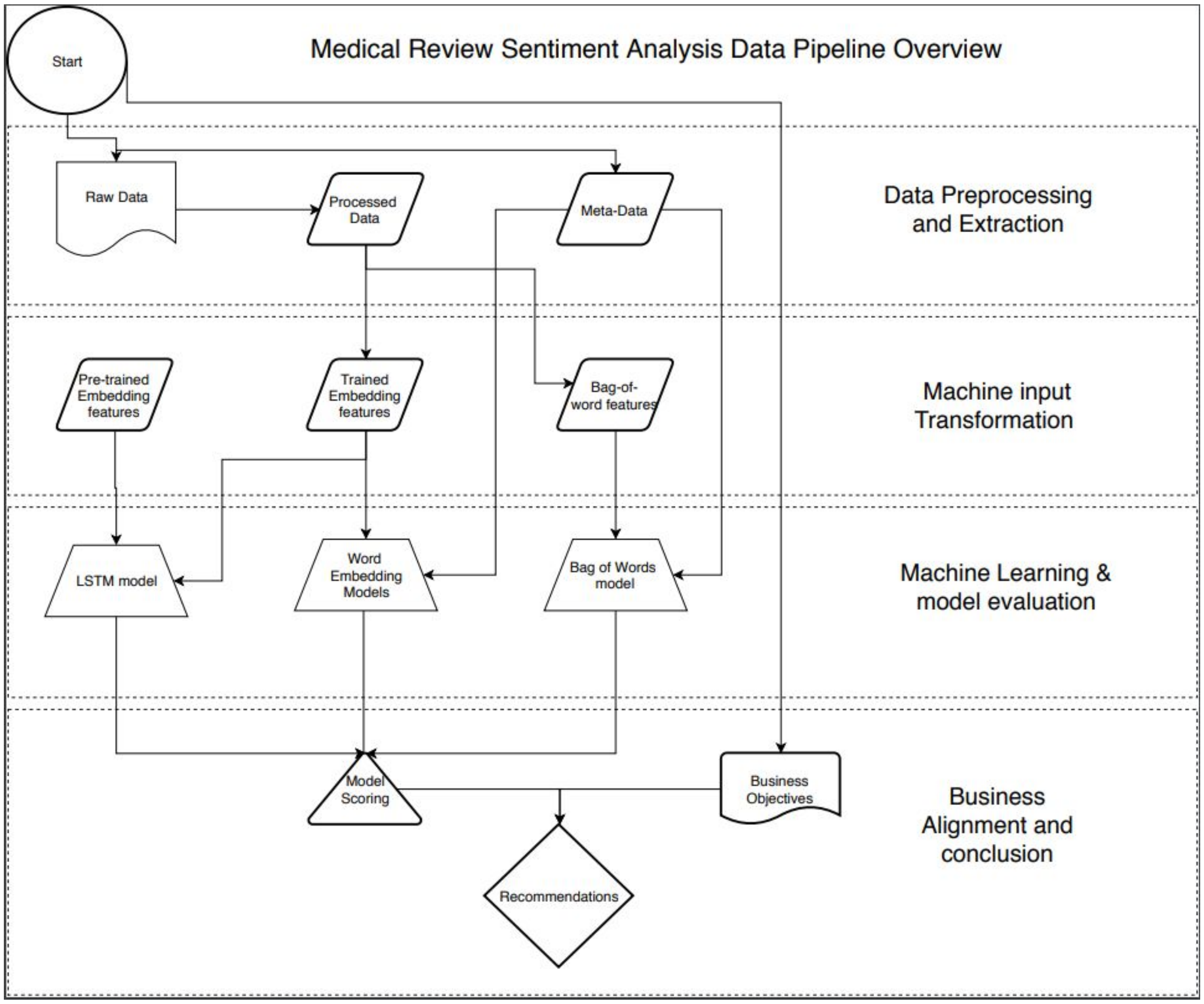
Once again, Kruskal-Wallis H test reinforces our belief with a p-value close to 0.

5. Word2Vec visualization: Due to the difficulty of this sentiment analysis task (it is difficult even for me, a human, to classify many of the sample reviews), we will want to know if our data is discriminative enough to provide value for the business. If not, the business is better off pursuing other opportunities. A few ways to visualize the discriminativeness of text data is to use WordCloud, LDA topic modelling and Word embeddings. This time, we will use TSNE dimensionality reduction technique to visualize word embeddings. To prevent clutter, I will use 2 words of each sentiment ('health', 'vaccine', 'effective', 'improve', 'bad', 'pain') and their 4 similar words to visualize the spread in two dimensions:



The result shows some distinctions. Positive words are on the middle and top left, whereas negative sentiments are in the bottom right. However, some neutral words are blurred with positive words, which could increase the possibility of misclassifications.

6. **Dataset conclusion and plan:** With the above analysis, it we were able to discover some discriminative features despite the foreseen difficulty in the classification task. The following plan is put forward as an attempt to build a valuable sentiment classification algorithm. In general, we will conduct simultaneously data processing and business alignment. On one hand, we will conduct sentiment analysis through our machine learning pipeline with the goal of finding the strongest all-around classification architecture measured by F1 Macro Score. On the other hand, we will consult with business personnel to analyze the cost-benefit of ML integration and build a Pay-Off matrix to translate the result of the algorithm into business values. In the end, we will tweek the best algorithm using custom architecture and make recommendations which can align with business objectives. The Overview of the project is outlined in the diagram below:


Medical Review Sentiment Analysis Data Pipeline Overview
Start
Raw Data
Processed Data
Meta-Data
Data Preprocessing and Extraction
Pre-trained Embedding features
Trained Embedding features
Bag-of-word features
Machine input Transformation
LSTM model
Word Embedding Models
Bag of Words model
Machine Learning & model evaluation
Model Scoring
Business Objectives
Business Alignment and conclusion
Recommendations

Machine Learning

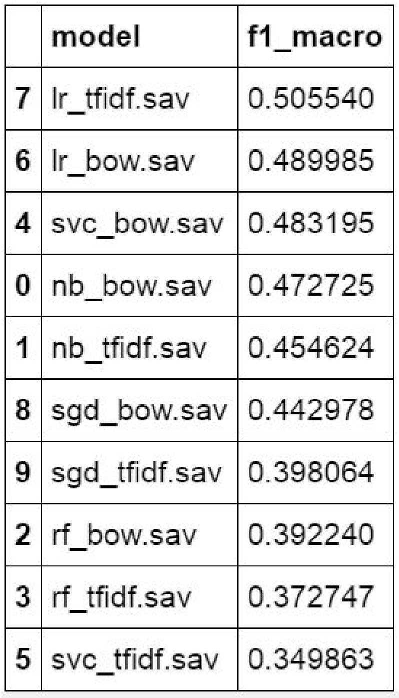
This section explore how we will run the data through our pipeline. Please refer to the pipeline diagram again for a clear process flow. The steps in the diagram are detailed as followed:

- With data pre-processing, we will preprocess the raw data and extract meta-data (which contains some discriminative information and can be useful as input for some of our machine learning algorithms).
- To extract readable features for our machine, we will use bag of words features, pretrained word embeddings and custom trained word-embeddings as our features. The data processed is transformed through the Bag of Word, Tfidf, Word2Vec, Matrix Factorization algorithms. Additionally, pre-trained word vectors from Google[2] (3 million vocabulary from webscrape) and Stanford[3] (400,000 vocabulary from wikipedia corpus) will be tried as our feature input.
- Next, we will run machine learning through cross validated grid-search on a variety of algorithms on our three types of feature input. For bag of word input features, the choice of algorithm is Logistic Regression (basic and simple parametric model), Naive Bayes (traditionally great on text andsparse data), Support Vector Machine and Linear Support Vector Machine with SGD training (great for high dimensional data), and Random Forest. We will choose the best model out of these to run the word embedding family feature input. Finally, we will build a recurrent neural network with LSTM layer to run on the text sequences with and without previous word embeddings.
- For evaluation, we will use F1 macro score (precision and recall in every class has equal weights) to address class imbalance in the dataset.

A. Bag of word models

We begin with running the aforementioned algorithms on the bag of word input, specifically the vanilla bag of word and the TFIDF version. TFIDF stands for Term Frequency Inverse Document Frequency and helps to the algorithms to filter out common words that exists throughout many documents and focus on the rare ones that can actually discriminate one sentiment from another. The results of the algorithms are summarized below:

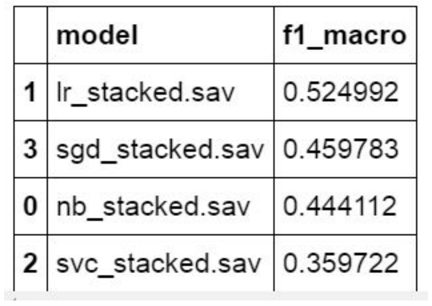
[2] Retrieved from: https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit
[3] Retrieved from: https://nlp.stanford.edu/projects/glove/



	model	f1_macro
7	lr_tfidf.sav	0.505540
6	lr_bow.sav	0.489985
4	svc_bow.sav	0.483195
0	nb_bow.sav	0.472725
1	nb_tfidf.sav	0.454624
8	sgd_bow.sav	0.442978
9	sgd_tfidf.sav	0.398064
2	rf_bow.sav	0.392240
3	rf_tfidf.sav	0.372747
5	svc_tfidf.sav	0.349863

Out of all the algorithms, Logistic Regression, especially the tfidf version, seems to pull ahead in terms of cross validation score. This is an expected result as logistic regression is a simple parametric approach that adds weights and is not prone to overfitting on small dataset. On the other hand, tree-based algorithms performed the worst due to the small number of samples and its tendency to overfit in sparse data (it's advantages are finding non-linear patterns). Support Vector Machine works very well with simple Bag of Word input but very poorly on TFIDF input. It seems that the normal bag of word approach allows the SVC to show its strengths in the high dimensionality of the data. Naive Bayes performs very fast and relatively well but it's not our favorite. A possible explanation for logistic regression's success is possibly the small sample size that the algorithms have to work with. It is possible that as we obtain more and more data, logistic regression will cease to improve. However, it is the best model we have to work with currently.

Stacking the bag of word feature spaces with meta-data obtained from pre-processing also improved logistic regression marginally. However, stacking meta-data into the input space hampered all the other algorithms as can be seen below.



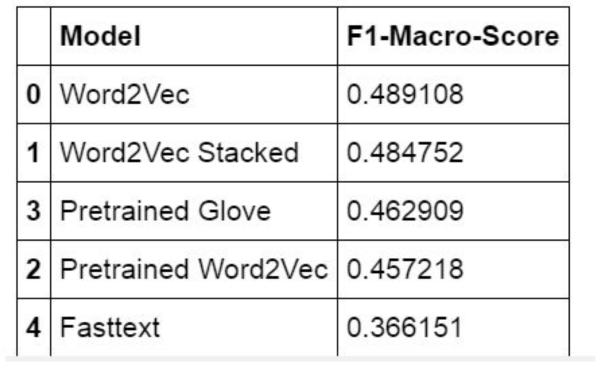
	model	f1_macro
1	lr_stacked.sav	0.524992
3	sgd_stacked.sav	0.459783
0	nb_stacked.sav	0.444112
2	svc_stacked.sav	0.359722

B. Word2Vec Embedding models

There are two weaknesses of the Bag-Of-Word approaches to sentiment analysis.

- First of all, BOW methods cannot handle comparison. If a document compares two products and say, for example, that Product 1 is better than Product 2, BOW will register a positive sentiment due to the word "better" regardless of whether the sentiment is about product 1 or 2. We can see that problem as there is a high confusion between Class 0 and Class 1 (positive and negative sentiment) than between Class 0 or 1 and Class 2 (neutral sentiment).
- Secondly, BOW methods don't fare so well for negations since they don't register semantics from words. Thus, "not terrible" will be seen as a negative sentiment solely because it has the word "terrible".

The Word2Vec family of input transformation addresses these two issues by adding a semantic layer which model the relationship between different words. Each word is represented with a number of dimensions that describe its distance with respect to other words. The distance can loosely be interpreted as some form of semantics. The result of running logistic regression on the word embedding input space (average embeddings of all words for a document) is outlined below:

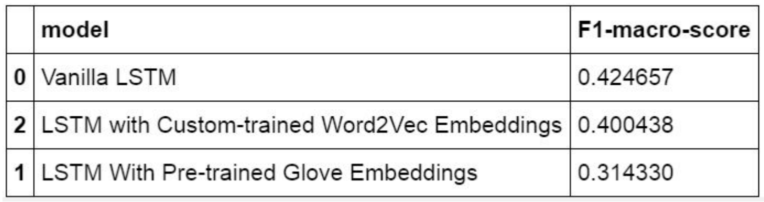


	Model	**F1-Macro-Score**
0	Word2Vec	0.489108
1	Word2Vec Stacked	0.484752
3	Pretrained Glove	0.462909
2	Pretrained Word2Vec	0.457218
4	Fasttext	0.366151

The vanilla custom-built word2vec embeddings input space works the best with a slightly lower score than the TFIDF input space in the previous section. This is arguably an improvement, as the model was able to get a good score with a dense matrix of only 200 dimensions, as oppose to the sparse matrix of before. Interestingly, stacking meta data into the input space cause the model to do worse. The pre-trained model from Google and Stanford wasn't too effective, most likely due to the fact that vocabularies the medical area is unlikely to be represented in a semantically similar way with other web corpus such as wikipedia.

C. LSTM with RNN

The final family of model we will experiment with is Neural Networks, more specifically, recurrent neural network (RNN). With this method, we can express the information in terms of sequence to extract another aspect of semantics (the ordering of words). Additionally, we'll be using Long Short Term Memory architecture for our RNN in order to help the model retain long-term information within a sequence and pre-trained word embeddings so that it can integrate other semantic aspects. The result of the models can be shown below.



	model	F1-macro-score
0	Vanilla LSTM	0.424657
2	LSTM with Custom-trained Word2Vec Embeddings	0.400438
1	LSTM With Pre-trained Glove Embeddings	0.314330

The Vanilla LSTM performs the best. A reason why it would perform better than with the Word2Vec embeddings is that its default embedding layers can learn more accurately. However, the RNN's performance pales in comparison to the algorithms we have explored so far. An explanation for this could be the small amount of sample size we have (most neural network requires 60,000 sample or more incomparison to approximately 4000 training example we have)