The impact of different pre-processing techniques on a deep learning-based opinion mining model for drug reviews

Pattarachai, Roongsritong (n10548467)
IFN704 Assessment 3

1. Executive Summary

Opinion mining allows us to extract more valuable insights on various fields including health care due to the explosive growth of social media. It often copes with textual data which is unstructured and has its own challenges. An opinion mining task includes pre-processing, feature engineering, model development, and evaluation. This project aims at studying the impact of pre-processing techniques with CNN-LSTM architecture on drug review data from the UCI machine learning repository. There were 4 models developed from different pre-processing techniques and one from a combination of the techniques. While the pre-processing step plays an integral part of the opinion mining task, there are a handful of studies focus on it, especially, for deep learning-based models and it still remains challenging. Moreover, there is a study reported that domain-specific data set such as medical data requires more sophisticated pre-processing techniques. The results suggest that all models achieved high accuracies and the model from lemmatisation gave the best performance; however, the combined model resulted in the poorest result among the models.

2. Introduction

In this era, the explosive growth of the Internet has become the main driving force of how people interact with each other. This results in a plethora of human expressions in various forms in the digital world including text, audio, images, and videos. There is an area of study related to computations of these expressions called opinion mining, which sometimes called sentiment analysis. This field enables us to effectively analyse how people think about any subject of interest which is an integral part of our decision-making process as people often seek others' opinions and take them into account prior to doing one.

Opinion mining can be done in a domain-specific manner as an opinionated word from one discourse might not reflect the same sentiment in another discourse [1]. In the pharmaceutical field, usually, products need to pass rigorous testing protocol to ensure safety for medical purposes. However, those standard protocols involve clinical trials under time and limited subject size constraints. Such conditions pose a potential risk of ADRs, as a result, post-sales product monitoring plays an integral role in maintaining medicine safety [2]. Additionally, opinion mining can also be utilised to build a health recommender system to reduce the workload of health care personals and potentially relieve their shortage.

In the past, the drug surveillance process relies heavily on structured data from clinics which is limited and requires substantial effort to prepare. Therefore, one of the potential solutions is to leverage opinion mining on drug user review data on the Internet [2]. Moreover, deep learning has recently gained its popularity from various disciplines including opinion mining due to the capabilities of dealing with large data and capturing non-linear relation between data [3] could help to deal with the data.

The impact of different pre-processing techniques on a deep learning-based opinion mining model for drug reviews

This project aims to find a way to enhance the opinion mining task with a deep learning-based model on drug review data via studying the impact of different text pre-processing techniques on the model's performance.

3. Literature Review

- ### The fundamental aspect of opinion mining

  Opinion mining inspects sentiment towards any entity or object mentioned in the text. The entity can consist of either components or a certain attribute such as 'this drug (entity) has terrible side effects (attribute). In opinion mining, there are three main classification levels based on how fine or coarse the text is considered in studies: document level, sentence level, and aspect-based level. The classification process includes text pre-processing, feature representation, classifier development, and evaluation. There is no doubt that the most important opinion indicators are opinion words such as good and bad which poses issues because they are context-dependent so they can have opposite orientations or polarities in different discourses [4]. As a result, text pre-processing is a crucial task since the balance between removing irrelevant information and keeping enough context must be maintained as well as to be studies domain specifically.

- ### Studies of text pre-processing in opinion mining

  Text pre-processing is the task of cleaning input textual data to remove unwanted information or noises because most online text reviews are written in formal languages. As a result, appropriate pre-processing steps can enhance the performance of opinion mining task. There are a number of text pre-processing techniques such as tokenisation, word stemming, lemmatisation, and part-of-speech tagging (POS).
  The pre-processed text then will be represented in machine-readable format prior to being fed into a classifying model. In spite of its importance, few studies have focused on pre-processing techniques particularly in neural-network models [5]. It is believed that the power of recent word embedding methods can generalise well enough and detect when different features carry similar information [6]. However, there are several studies [7]–[12] conducted on traditional machine learning algorithms such as Support Vector Machine (SVM) and suggested that a proper and combined technique showed significant improvement in models' accuracies.
  There is a study of finding the impact of text pre-processing on deep learning models for sentiment analysis which was done by using tokenisation, lowercasing, lemmatisation, and multiword grouping [5]. It showed that in a domain-specific data set especially the medical data, more sophisticated text-pre-processing techniques (lemmatisation and multiword grouping) outperformed simple techniques (only tokenisation and only lowercasing) and the study recommended future attempts to study more techniques such as stopword removal. Therefore, it is worth investigating the impact of more text pre-processing techniques in deep learning-based models in domain-specific data such as drug reviews. Additionally, [4] suggests that selecting effective text-pre-processing still remains a challenging task.

- ### Opinion Classifying Techniques

  Despite a range of sentiment classifying models proposed by researchers, they can be grouped into 3 main categories: supervised, unsupervised, and deep learning[4]. Because of its benefit of learning new features from training data, the deep learning approach is more efficient in providing us with more accurate results in opinion mining task than the traditional machine learning models [3], [4], [13]. According to the review articles of [3], [13], Convolutional Neural Network (CNN) and Long Short-term Memory Network (LSTM) are popular models which showed high accuracy on various data sets. In addition to this, [14], [15] could improve performance of opinion mining task by combining both CNN and LSTM architectures where the former network learns local features from the text and the latter detects long-term dependencies from each sentence.

- Conclusion

  Text pre-processing is important yet challenging in opinion mining task and there are some rooms for exploring its role in deep learning approaches, especially, in domain specific such as drug review data. In this project, the impact of different pre-processing techniques on a state-of-the-art deep learning model (CNN with LSTM) was investigated for document level opinion mining task on a drug review data set. The model can tell whether an input text is either positive or negative which has an application in preliminary drug treatment quality assessment.

4. Approach

The investigation of this work was addressed by making use of artefact-oriented research methodology and empirical comparison. A drug review data set was pre-processed with tokenisation, which parsing text into small units called 'token', and represented features by Word2Vec algorithm before developing a baseline model with CNN and LSTM architecture. After that other text pre-processing techniques: stopword removal, stemming, lemmatisation, combined method were applied on the data and fed into the baseline model. Finally, all 5 models were evaluated by performance metrices. The high-level overview of the project pipeline is shown in figure 1.

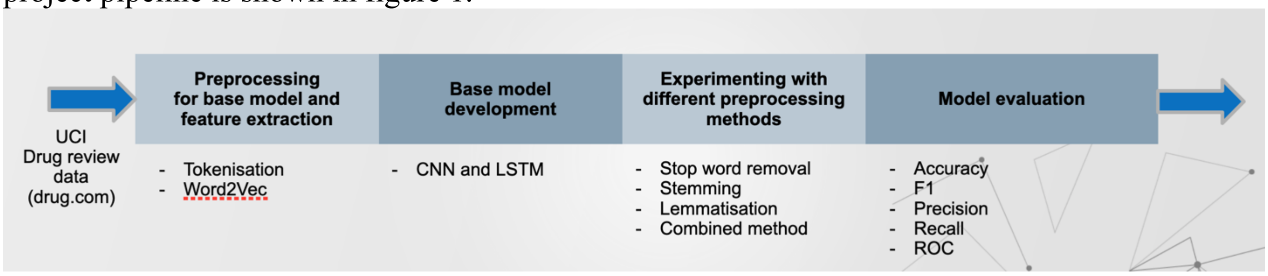


Figure 1. This project pipeline

4.1 Data

The data used in this work was made available by [2] on the UCI machine learning repository. It was originally retrieved by web crawling from Drug.com which is a medicine review website providing textual reviews from specific drugs with related conditions and 10-scaled numerical reviews. The data was pre-split with ratio of 75/25 for training and test sets, respectively. The data dictionary is demonstrated in the table below.

Field Name	Data Type	Description
drugName	categorical	The name of the drug
condition	categorical	A name of the condition to use the drug
review	text	A review from a patient

rating	numerical	10-scale rating
date	date	Date of review entry
usefulCount	numerical	The number of users who found the review useful

Table 1. Data dictionary of reviews on Drug.com

4.2 Exploratory Analysis

In order to get a good grasp of the data set, some descriptive analysis was performed on the data. According to figure 2, the rating attribute of the data was plotted with histograms. There are 161,297 entries in the training set and 53,766 entries for the test set. It shows that the majority of reviews are highly rated in both training and test sets.

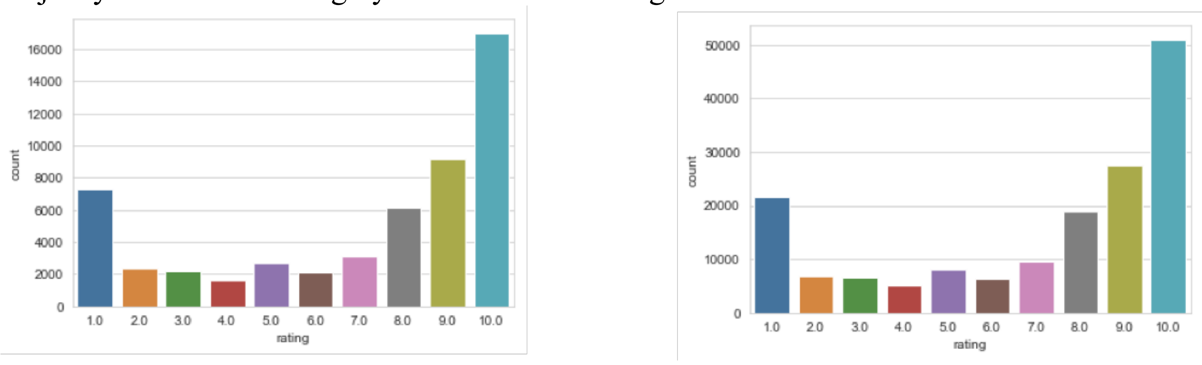


Figure 2. Histograms of rating attributes for training (left) and test (right) sets

For better visualisation and preparation for the model development, the data sets were combined, and the rating attribute was made into two classes: positive (1) for the ratings more than 5, and negative (0) for the rest as shown in figure 3. Furthermore, it can be noticed that the class imbalance is obvious.

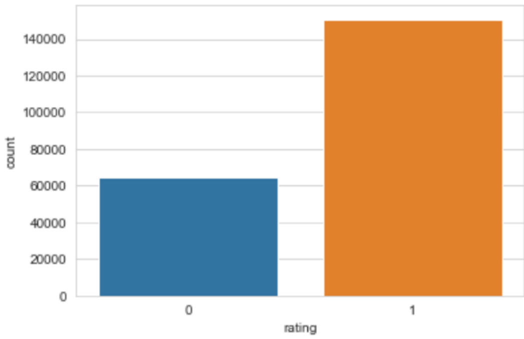


Figure 3. Rating attribute of the combined data set

The Word Cloud technique was also used to see most frequent words in each class as in figure 4.

Figure 4. Word Cloud for positive (left) and negative (right) classes

It appears that they did not indicate much about the difference between each class because there are several words in common in both classes. Additionally, the review text was used to find the polarity of each review by TextBlob library. The reviews with the top-3 highest (positive) and lowest (negative) polarities are shown in figure 5 which seems to be reasonable as there are words such as 'excellent' in high polarity review and 'terrible' in low polarity review.

```
3 Random Reviews with Highest Polarity:
Review 1:
 "Off and on all my life I&#039;ve been bothered by Yeast infections and have never found anything better than the Va
gistat. Esp. since it&#039;s a one time treatment and you&#039;re done with it.  BUT also the fact that it&#039;s not
only beneficial for the Candida strain but also some of the rarer resistant types of yeast that are often missed. It
might irritate for that day but it works and well worth it in my opinion. Thanks for letting me have some input.
D.T."
Review 2:
 "Best medicine I ever seen."
Review 3:
 "It has been excellent in curing my symptoms."
```

```
3 Random Reviews with Lowest Polarity:
Review 1:
 "I took this for restless leg syndrome and I had terrible involuntary muscle spasms and leg and arm movement. I had
to go to the ER. It was terrible."
Review 2:
 "So I&#039;ve had the implant since June 2015 and I haven&#039;t stopped bleeding it&#039;s so annoying the reason I
got it was so I wouldn&#039;t worry about getting a period every month but I&#039;ve been on my period since I got it
! idk whether to leave it or just take it off ."
Review 3:
 "Drug works but the patch is horrible.  It is plastic and wrinkles up.  The edges don&#039;t adhere well so the patc
h is always sticking to my clothes.  I hope they fix this issue.  If they do I will buy again but for now I&#039;m go
ing to save my money."
```

Figure 5. Review Text polarity with TextBlob

Subsequently, as mentioned above, the data set is imbalanced by having an unequal ratio of the target classes. This can cause issues in classification task as developed models can experience overfitting in the majority class. Therefore, to solve such an issue, the Synthetic Minority Over-sampling Technique (SMOTE) was used in this work. The technique increases the number of the minority class in the ratio of majority class which could be helpful in model development as [16] reported to overcome the class imbalance issue on Twitter sentiment analysis by mean of SMOTE. As a result, the data set was class balanced and downsized to 5% (roughly 15,000 entries) to cope with computation expenses via stratified sampling.

4.3 Baseline Model Development

- Text pre-processing
  Tokenisation is the simplest text pre-processing technique which is also required in other techniques. This technique splits the whole text into smaller units such as characters, words, or sub-words and makes it less difficult to transform into a machine-readable format.

- Feature Extraction
  After tokenisation, the tokens were converted into numerical representation for a machine to read. The Word2Vec algorithm was reported to be an effective method for both traditional and deep learning-based sentiment analysis [9], [17], [18]. It was originally proposed by [19] and the main idea of the algorithm is to use neural networks to learn word associations from a large text corpus and represent words in a vector space. The algorithm can be used to build a pre-trained word embedding layer for deep-learning models. In this project, a pre-trained word embedding from the PubMed Central Open Access Case Report as cited in [20] was used to make sure that

Word2Vec learns and captures feature properly in the medical context. This enables machines to work with text such as find word similarity in figure 6.

```
model.wv.most_similar('pain')#, topn =1)

[('pain_and', 0.9086359739303589),
 ('pain_intensity', 0.8848637938499451),
 ('acute_pain', 0.8739394545555115),
 ('ongoing_pain', 0.8737969994544983),
 ('persistent_pain', 0.8737456798553467),
 ('pain,', 0.8705431818962097),
 ('pain_or', 0.8694646954536438),
 ('chronic_pain', 0.8577300906181335),
 ('intensity_of_pain', 0.8502391576766968),
 ('back_pain', 0.8502222895622253)]
```

Figure 6. An example of finding word similarity from the Word2Vec model

- ## Model Development

The data from the previous step was split into training, validation (for hyperparameter tuning), and test (unseen) sets with ratios of 70%, 20%, and 10%, respectively. A combination of CNN and LSTM architectures were used by having the independent variable as the pre-processed text and the response variable as the target class rating with the pre-trained embedding layer from Word2Vec. After spending some time on manual tuning, a model with acceptable accuracy and no sign of overfitting after 15 epochs shown in figure 7 was obtained with the hyperparameter settings as in figure 8.

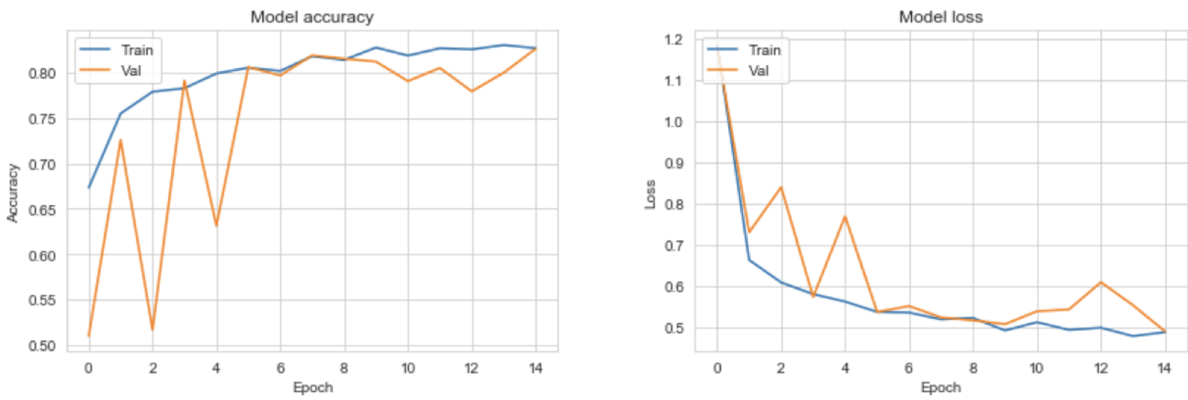


Figure 7. Model training curves

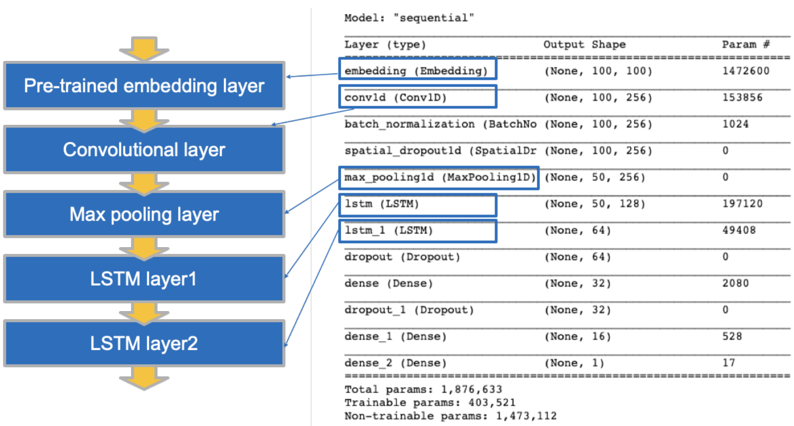


Figure 8. The obtained model settings

In addition to training accuracy and overfitting, a confusion matrix and ROC curve were also used to determine how good the model was. In figure 9, it shows that the model was performing a good classification; however, there was a concern about the false positive as patients who have negative feedback should not be neglected and they might need medical attention. This makes false positive rate is one of the evaluation matrices to consider model performance. On the right of figure 9, the ROC curve shows that the area under the curve was 0.904 which is relatively high. As a result, the model architecture and hyperparameters were kept as a baseline.

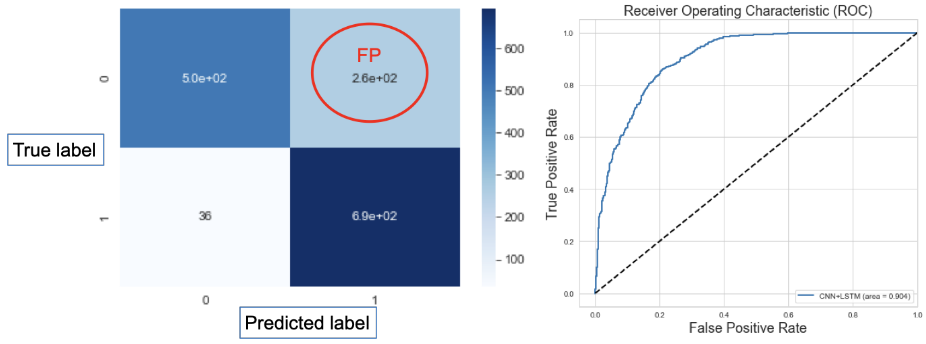


Figure 9. Confusion matrix and ROC curve of the baseline model

4.4 Experimenting with Different Pre-processing Methods

At this stage, other text pre-processing techniques were applied on top of the tokenised data prior to executing the feature extraction to build more models using the same hyperparameter settings as the baseline model. There are three other pre-processing techniques used in this work as follows:

stopword removal – removing frequently appeared words that do not carry much information. This project used a stopword list from the NLTK library.

Word stemming – reducing word forms by mainly truncating prefixes and suffixes. Although there are various stemming algorithms, this work used Porter Stemming which is commonly used in natural language processing.

Lemmatisation – also reducing word forms but by mainly converting a word into its lemma (root). The WordNet Lemmatizer was used in this work.

To highlight the difference between stemming and lemmatization, figure 10 is shown and lemmatisation is more complex than stemming.

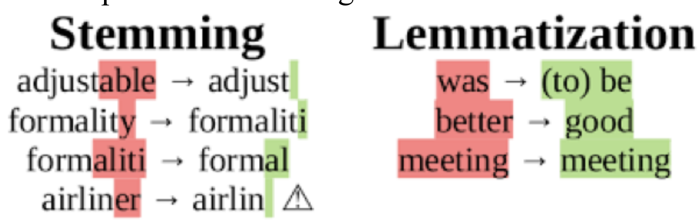


Figure 10. Difference of stemming and lemmatisation (source: https://devopedia.org/lemmatization)

Additionally, a combination of the three methods was also used to build a model.

4.5 Model Evaluation

Finally, all models were evaluated by calculating common evaluation matrics for sentiment analysis [3] from the test data set as in table 2: accuracy, precision, recall, F1-score, area under ROC as well as their confusion metrics.

Model	Accuracy	Precision	Recall	F1	ROC
Base Model	0.8	0.73	0.95	0.82	0.9
Stop Word Removal	0.78	0.71	0.91	0.8	0.87
Stemming	0.79	0.75	0.86	0.8	0.88
Lemmatisation	0.82	0.77	0.88	0.82	0.9

Combined Model	0.74	0.72	0.77	0.74	0.84

Table 2. Evaluation metrics of each model

5. Findings

To make comparison easier, table 2 was plotted using the bar chart.

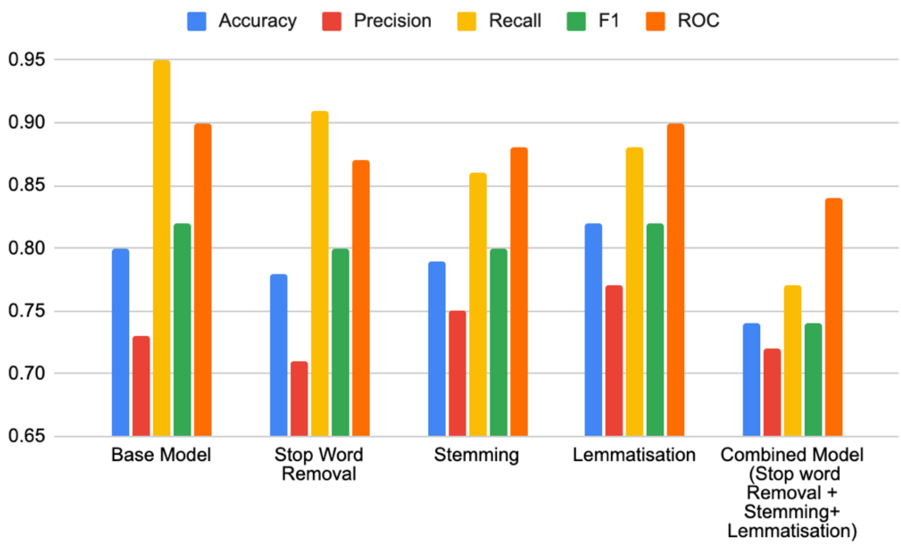


Figure 11. A bar chart of evaluation metrics of each model

Figure 11 demonstrates that the combination of CNN and LSTM architecture gave a relatively high performance on the drug review data set, especially, the baseline model outperformed stopword removal, stemming, and the combined model which was unexpected and models with different pre-processing techniques resulted in different results. This could be because stopword removal and word stemming either removed or parsed some words in a way that destroyed the context of review text which plays an integral part when a deep learning-based model learns the features. Apart from this, the model with lemmatisation seems to be the best performing model on this data set not only by the overall results of evaluation matrics but the false positive rate according to the confusion matrix also highlights that which resembles what [5] found on another medical data set with a deep learning-based model as well.

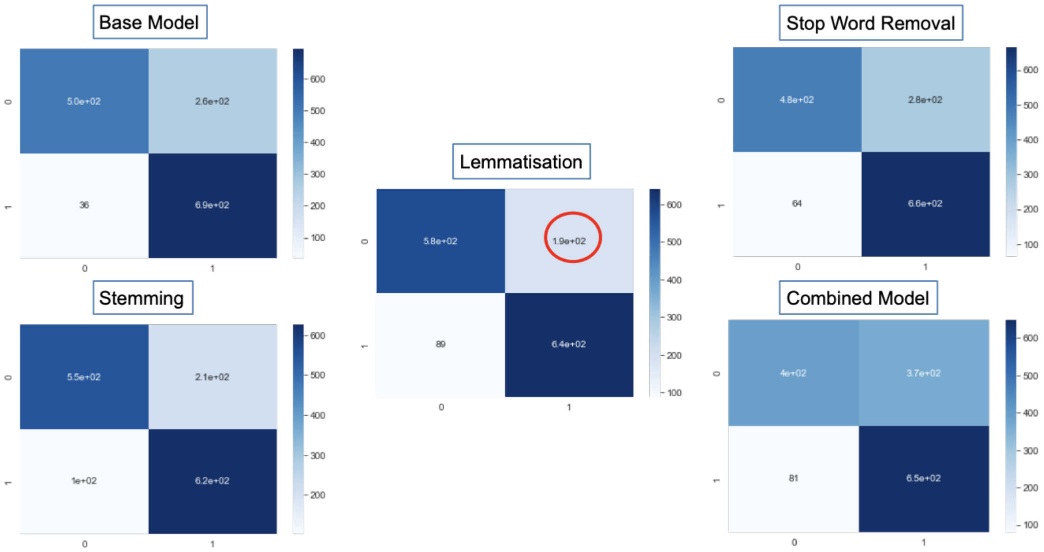


Figure 12. Confusion matrices of all models

Nevertheless, the combined model has the poorest result in contrast to [9], [10], and [12] who claimed that combined techniques should perform better on other non-domain specific data sets with traditional machine learning models. This potentially indicates that some techniques are not appropriate to use together at the same time. In the combined model, word stemming, and lemmatisation could be redundant to each other since they have the same job to reduce word forms. In addition to this, the differences of most models are subtle possibly because the pre-trained word embedding generalised so well that compensate for the effect of pre-processing approaches.

6. Reflection

During this study, I felt like to had to climb a learning curve all the time as I had no experience working with textual data; however, I found it challenging and there was an opportunity to learn as it could be a favour to add a new skill on my toolbelt to secure a job in the future. I also had to get through stressful time from my available resources without being able to go to the campus from overseas since working with large-scale data and deep learning are computationally expensive. At first, each training epoch took more than 10 minutes so that I gave up and inevitably chose to work with downsized data which was still required substantial time to run.

In summary, this work found that the CNN and LSTM model with Word2Vec pre-trained word embedding could achieve high performance on the drug review data set in the opinion mining task. In addition to this, text pre-processing techniques can affect model performance so proper technique should be used wisely, especially, if one were to use a combination of techniques to achieve a better result. It appeared that, in this project, the model with lemmatisation resulted in the best performance.

Importantly, there are some limitations in this work as only one model architecture was investigated on a single data set so it cannot generalise either to other deep learning-based models or other drug review data sets. Additionally, the pre-trained embedding was trained on medical case reports in a clinical setting so it might not reflect how patients give their feedbacks to professionals. In future work, it is recommended to work on more data amount on the same data set as well as other similar domain data sets with more deep learning architectures. More text pre-processing techniques should also be investigated with a range of combinations. The GridSearch algorithm is a good choice to replace manual hyperparameter tuning. Additionally, the evaluation results could be made more reliable with cross-validation and t-test on results; however, most recommendations for future work requires more powerful computing resources to execute.

References

[1] K. Denecke, 'Sentiment Analysis from Medical Texts', in *Health Web Science*, Cham: Springer International Publishing, 2015, pp. 83–98. doi: 10.1007/978-3-319-20582-3_10.

[2] F. Gräßer, S. Kallumadi, H. Malberg, and S. Zaunseder, 'Aspect-Based Sentiment Analysis of Drug Reviews Applying Cross-Domain and Cross-Data Learning', in *Proceedings of the 2018 International Conference on Digital Health*, Lyon France, Apr. 2018, pp. 121–125. doi: 10.1145/3194658.3194677.

[3] A. Yadav and D. K. Vishwakarma, 'Sentiment analysis using deep learning architectures: a review', *Artif. Intell. Rev.*, vol. 53, no. 6, pp. 4335–4385, Aug. 2020, doi: 10.1007/s10462-019-09794-5.

[4] L. D. C. S. Subhashini, Y. Li, J. Zhang, A. S. Atukorale, and Y. Wu, 'Mining and classifying customer reviews: a survey', *Artif. Intell. Rev.*, Mar. 2021, doi: 10.1007/s10462-021-09955-5.

[5] J. Camacho-Collados and M. T. Pilehvar, 'On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis', *ArXiv170701780 Cs*, Aug. 2018, Accessed: Mar. 27, 2021. [Online]. Available: http://arxiv.org/abs/1707.01780

[6] J. Camacho-Collados and M. T. Pilehvar, 'From Word to Sense Embeddings: A Survey on Vector Representations of Meaning', *ArXiv180504032 Cs*, Oct. 2018, Accessed: Mar. 27, 2021. [Online]. Available: http://arxiv.org/abs/1805.04032

[7] A. K. Uysal and S. Gunal, 'The impact of preprocessing on text classification', *Inf. Process. Manag.*, vol. 50, no. 1, pp. 104–112, Jan. 2014, doi: 10.1016/j.ipm.2013.08.006.

[8] R. Duwairi and M. El-Orfali, 'A study of the effects of preprocessing strategies on sentiment analysis for Arabic text', *J. Inf. Sci.*, vol. 40, no. 4, pp. 501–513, Aug. 2014, doi: 10.1177/0165551514534143.

[9] S. Alam and N. Yao, 'The impact of preprocessing steps on the accuracy of machine learning algorithms in sentiment analysis', *Comput. Math. Organ. Theory*, vol. 25, no. 3, pp. 319–335, Sep. 2019, doi: 10.1007/s10588-018-9266-8.

[10] S. Pradha, M. N. Halgamuge, and N. Tran Quoc Vinh, 'Effective Text Data Preprocessing Technique for Sentiment Analysis in Social Media Data', in *2019 11th International Conference on Knowledge and Systems Engineering (KSE)*, Da Nang, Vietnam, Oct. 2019, pp. 1–8. doi: 10.1109/KSE.2019.8919368.

[11] A. Krouska, C. Troussas, and M. Virvou, 'The effect of preprocessing techniques on Twitter sentiment analysis', in *2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA)*, Chalkidiki, Greece, Jul. 2016, pp. 1–5. doi: 10.1109/IISA.2016.7785373.

[12] D. Effrosynidis, S. Symeonidis, and A. Arampatzis, 'A Comparison of Pre-processing Techniques for Twitter Sentiment Analysis', in *Research and Advanced Technology for Digital Libraries*, vol. 10450, J. Kamps, G. Tsakonas, Y. Manolopoulos, L. Iliadis, and I. Karydis, Eds. Cham: Springer International Publishing, 2017, pp. 394–406. doi: 10.1007/978-3-319-67008-9_31.

[13] L. Zhang, S. Wang, and B. Liu, 'Deep learning for sentiment analysis: A survey', *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 8, no. 4, Jul. 2018, doi: 10.1002/widm.1253.

[14] Q. Huang, R. Chen, X. Zheng, and Z. Dong, 'Deep Sentiment Representation Based on CNN and LSTM', in *2017 International Conference on Green Informatics (ICGI)*, Fuzhou, Aug. 2017, pp. 30–33. doi: 10.1109/ICGI.2017.45.

[15] A. Hassan and A. Mahmood, 'Deep Learning approach for sentiment analysis of short texts', in *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*, Nagoya, Japan, Apr. 2017, pp. 705–710. doi: 10.1109/ICCAR.2017.7942788.

[16] K. Ghosh, A. Banerjee, S. Chatterjee, and S. Sen, 'Imbalanced Twitter Sentiment Analysis using Minority Oversampling', in *2019 IEEE 10th International Conference on Awareness Science and Technology (iCAST)*, Morioka, Japan, Oct. 2019, pp. 1–5. doi: 10.1109/ICAwST.2019.8923218.

[17] M. K. Sohrabi and F. Hemmatian, 'An efficient preprocessing method for supervised sentiment analysis by converting sentences to numerical vectors: a twitter case study', *Multimed. Tools Appl.*, vol. 78, no. 17, pp. 24863–24882, Sep. 2019, doi: 10.1007/s11042-019-7586-4.

[18] Y. Goldberg, 'A Primer on Neural Network Models for Natural Language Processing', *ArXiv151000726 Cs*, Oct. 2015, Accessed: Mar. 27, 2021. [Online]. Available: http://arxiv.org/abs/1510.00726

[19] T. Mikolov, K. Chen, G. Corrado, and J. Dean, 'Efficient Estimation of Word Representations in Vector Space', *ArXiv13013781 Cs*, Sep. 2013, Accessed: Mar. 27, 2021. [Online]. Available: http://arxiv.org/abs/1301.3781

[20] Z. N. Flamholz, L. H. Ungar, and G. E. Weissman, 'Word embeddings trained on published case reports are lightweight, effective for clinical tasks, and free of protected health information', Health Informatics, preprint, Dec. 2019. doi: 10.1101/19013268.

Appendix 1: Supplementary Information

A source code in Python which supports this project analysis can be accessed via the link below.
https://drive.google.com/file/d/18GZexO6UCXEive5Xfaq-W_yy6BvHXjiX/view?usp=sharing

Alternatively, a markdown version of the source code is shown as follows.

In [2]:

```
# from google.colab import drive
# drive.mount('/content/drive')
```

https://github.com/gweissman/clinical_embeddings

In [3]:

```
import pandas as pd
import numpy as np
import seaborn as sns
```

Training Set

In [4]:

```
df_train = pd.read_csv("/Users/stamp/Desktop/Semester1-2021/IFN704/data/drugsCom_raw/drugsComTrain_raw.ts
df_train.head()
```

Out[4]:

	Unnamed: 0	drugName	condition	review	rating	date	usefulCount
0	206461	Valsartan	Left Ventricular Dysfunction	"It has no side effect, I take it in combinati...	9.0	May 20, 2012	27
1	95260	Guanfacine	ADHD	"My son is halfway through his fourth week of ...	8.0	April 27, 2010	192
2	92703	Lybrel	Birth Control	"I used to take another oral contraceptive, wh...	5.0	December 14, 2009	17
3	138000	Ortho Evra	Birth Control	"This is my first time using any form of birth...	8.0	November 3, 2015	10
4	35696	Buprenorphine / naloxone	Opiate Dependence	"Suboxone has completely turned my life around...	9.0	November 27, 2016	37

In [5]:

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 161297 entries, 0 to 161296
Data columns (total 7 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   Unnamed: 0   161297 non-null  int64  
 1   drugName     161297 non-null  object 
 2   condition    160398 non-null  object 
 3   review       161297 non-null  object 
 4   rating       161297 non-null  float64
 5   date         161297 non-null  object 
 6   usefulCount  161297 non-null  int64  
dtypes: float64(1), int64(2), object(4)
memory usage: 8.6+ MB
```

In [6]:

```
# drop the columns that are not relevant
df_train = df_train.drop(columns=['Unnamed: 0', 'drugName', 'condition', 'date', 'usefulCount'])
```

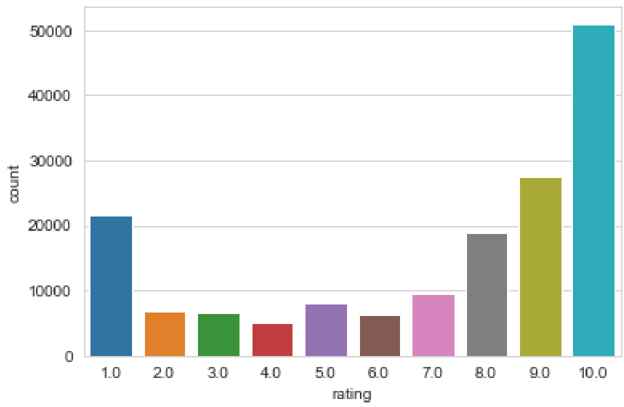
In [7]:

```
target_train = df_train['rating']
sns.set_style('whitegrid')
sns.countplot(target_train)
```

```
/Users/stamp/opt/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[7]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa971129a90>
```



In [8]:

```python
# Giving the Sentiment according to their ratings
df_train['rating'] = df_train['rating'].apply(lambda x: 1 if x > 5 else 0)
df_train.head()
```

Out[8]:

	review	rating
0	"It has no side effect, I take it in combinati...	1
1	"My son is halfway through his fourth week of ...	1
2	"I used to take another oral contraceptive, wh...	0
3	"This is my first time using any form of birth...	1
4	"Suboxone has completely turned my life around...	1

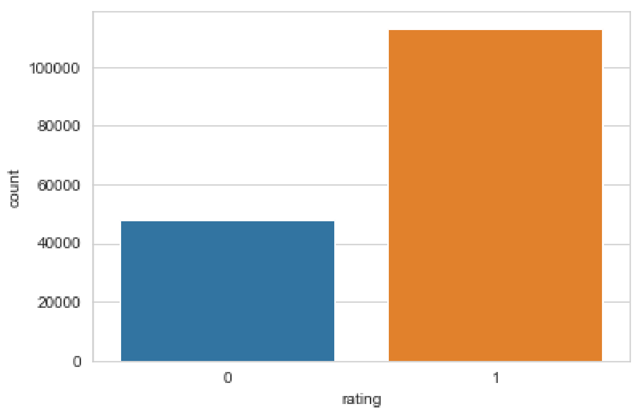
In [9]:

```python
sns.set_style('whitegrid')
sns.countplot(df_train['rating'])
```

```
/Users/stamp/opt/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa95a4926d0>
```



Test Set

In [10]:

```python
df_test = pd.read_csv("/Users/stamp/Desktop/Semester1-2021/IFN704/data/drugsCom_raw/drugsComTest_raw.tsv'
df_test.head()
```

Out[10]:

	Unnamed: 0	drugName	condition	review	rating	date	usefulCount
0	163740	Mirtazapine	Depression	"I've tried a few antidepressants over th...	10.0	February 28, 2012	22
1	206473	Mesalamine	Crohn's Disease, Maintenance	"My son has Crohn's disease and has done ...	8.0	May 17, 2009	17
2	159672	Bactrim	Urinary Tract Infection	"Quick reduction of symptoms"	9.0	September 29, 2017	3
3	39293	Contrave	Weight Loss	"Contrave combines drugs that were used for al...	9.0	March 5, 2017	35
4	97768	Cyclafem 1 / 35	Birth Control	"I have been on this birth control for one cyc...	9.0	October 22, 2015	4

In [ ]:

df_test.info()

In [11]:

```
# drop the columns that are not relevant
df_test = df_test.drop(columns=['Unnamed: 0', 'drugName', 'condition', 'date', 'usefulCount'])
```

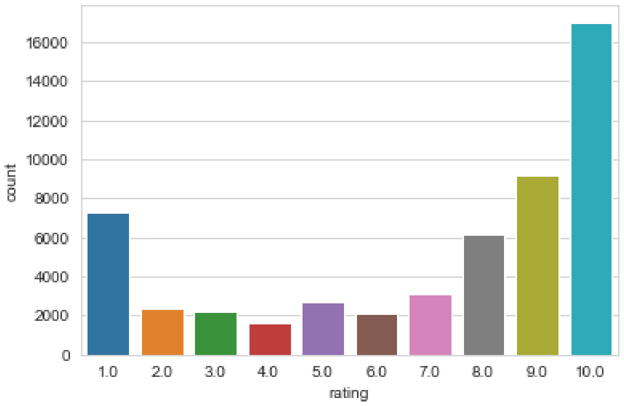
In [12]:

```
target_test = df_test['rating']
sns.set_style('whitegrid')
sns.countplot(target_test)
```

```
/Users/stamp/opt/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass
the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will
be `data`, and passing other arguments without an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
```

Out[12]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa95d499c10>
```



In [13]:

```
# Giving the Sentiment according to their ratings
df_test['rating'] = df_test['rating'].apply(lambda x: 1 if x > 5 else 0)
df_test.head()
```

Out[13]:

	review	rating
0	"I've tried a few antidepressants over th...	1
1	"My son has Crohn's disease and has done ...	1
2	"Quick reduction of symptoms"	1
3	"Contrave combines drugs that were used for al...	1
4	"I have been on this birth control for one cyc...	1

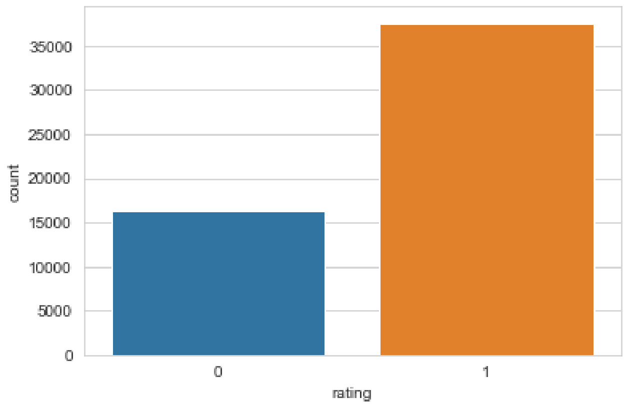
In [14]:

```
sns.set_style('whitegrid')
sns.countplot(df_test['rating'])
```

```
/Users/stamp/opt/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa95d4993d0>
```



In [15]:

```
# X_train = df_train.loc[:, 'review'].values
# y_train = df_train.loc[:, 'rating'].values
# X_test = df_test.loc[:, 'review'].values
# y_test = df_test.loc[:, 'rating'].values
```

Combine the two data sets

In [16]:

```
df = df_train.append(df_test)
df.head()
```

Out[16]:

	review	rating
0	"It has no side effect, I take it in combinati...	1
1	"My son is halfway through his fourth week of ...	1
2	"I used to take another oral contraceptive, wh...	0
3	"This is my first time using any form of birth...	1
4	"Suboxone has completely turned my life around...	1

In [17]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 215063 entries, 0 to 53765
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype 
---  ------  --------------   ----- 
 0   review  215063 non-null  object
 1   rating  215063 non-null  int64 
dtypes: int64(1), object(1)
memory usage: 4.9+ MB
```

Plot bar chart 0 and 1 class + wordcloud + top 10 for each class

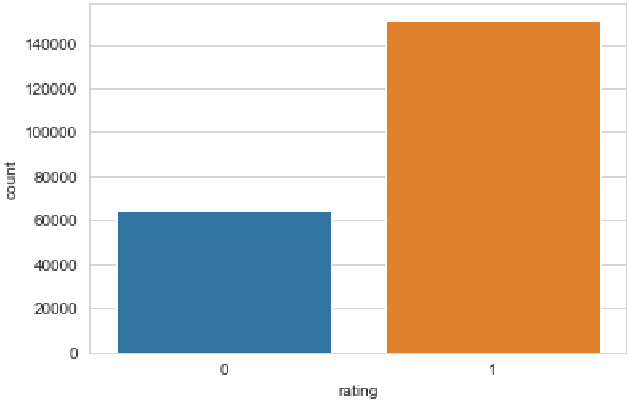
In [18]:

```
sns.set_style('whitegrid')
sns.countplot(df['rating'])
```

```
/Users/stamp/opt/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[18]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa9715ca750>
```



In [19]:

```
pip install wordcloud
```

```
Requirement already satisfied: wordcloud in /Users/stamp/opt/anaconda3/lib/python3.7/site-packages (1.8.1)
Requirement already satisfied: numpy>=1.6.1 in /Users/stamp/opt/anaconda3/lib/python3.7/site-packages (from wordcloud) (1.19.5)
Requirement already satisfied: matplotlib in /Users/stamp/.local/lib/python3.7/site-packages (from wordcloud) (3.0.2)
Requirement already satisfied: pillow in /Users/stamp/opt/anaconda3/lib/python3.7/site-packages (from wordcloud) (8.1.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /Users/stamp/.local/lib/python3.7/site-packages (from matplotlib->wordcloud) (1.0.1)
Requirement already satisfied: cycler>=0.10 in /Users/stamp/.local/lib/python3.7/site-packages (from matplotlib->wordcloud) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in /Users/stamp/.local/lib/python3.7/site-packages (from matplotlib->wordcloud) (2.8.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /Users/stamp/.local/lib/python3.7/site-packages (from matplotlib->wordcloud) (2.3.1)
Requirement already satisfied: six in /Users/stamp/opt/anaconda3/lib/python3.7/site-packages (from cycler>=0.10->matplotlib->wordcloud) (1.15.0)
Requirement already satisfied: setuptools in /Users/stamp/opt/anaconda3/lib/python3.7/site-packages (from kiwisolver>=1.0.1->matplotlib->wordcloud) (52.0.0.post20210125)
Note: you may need to restart the kernel to use updated packages.
```

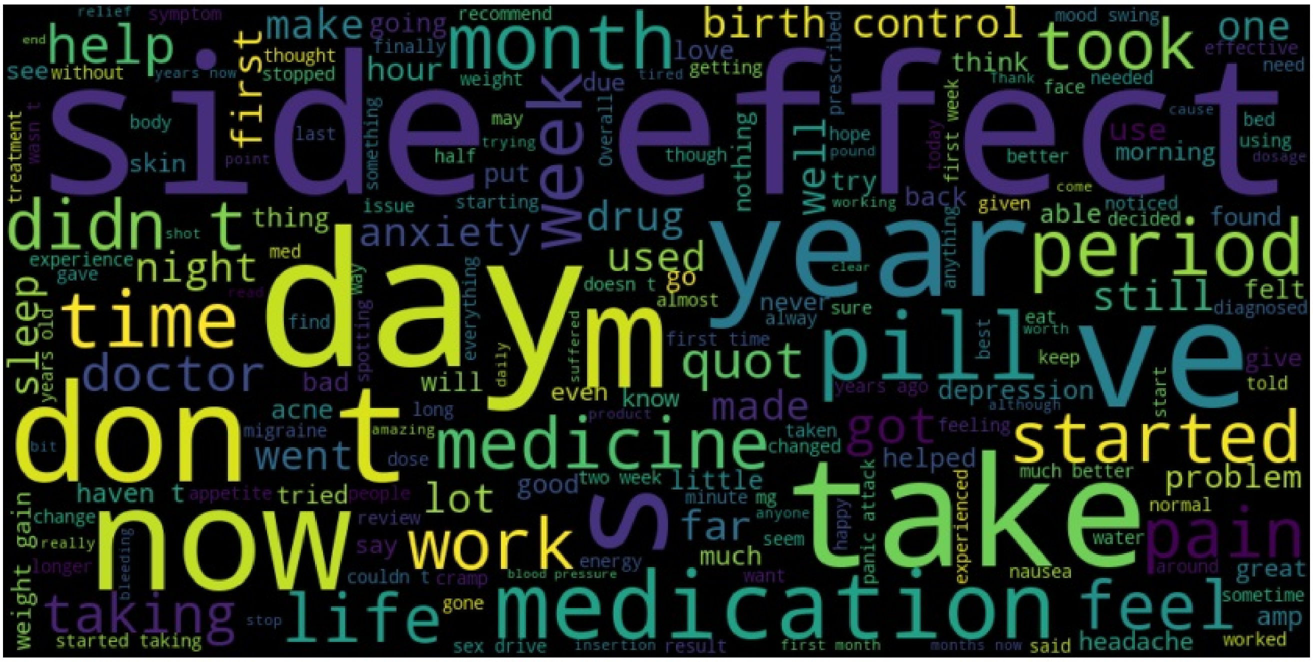
Word Cloud Positive

In [20]:

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt

df_rate_positive = df.loc[df.rating == 1, 'review']
k1 = (' '.join(df_rate_positive))

wordcloud = WordCloud(width = 1000, height = 500).generate(k1)
plt.figure(figsize=(15, 10))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off');
```

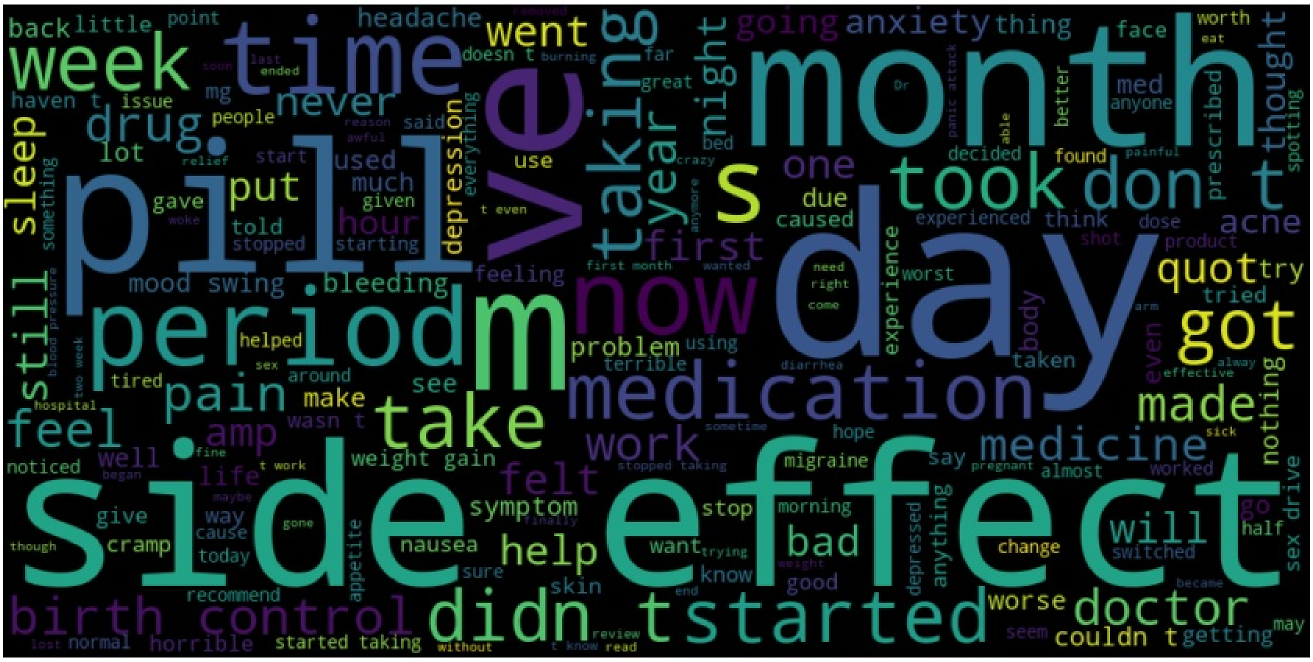


Word Cloud Negative

In [21]:

```
df_rate_negative = df.loc[df.rating == 0, 'review']
k1 = (' '.join(df_rate_negative))

wordcloud = WordCloud(width = 1000, height = 500).generate(k1)
plt.figure(figsize=(15, 10))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off');
```



In [22]:

```
pip install textblob
```

```
Requirement already satisfied: textblob in /Users/stamp/opt/anaconda3/lib/python3.7/site-packages (0.15.3
)
Requirement already satisfied: nltk>=3.1 in /Users/stamp/opt/anaconda3/lib/python3.7/site-packages (from
textblob) (3.5)
Requirement already satisfied: tqdm in /Users/stamp/opt/anaconda3/lib/python3.7/site-packages (from nltk>
=3.1->textblob) (4.56.0)
Requirement already satisfied: joblib in /Users/stamp/opt/anaconda3/lib/python3.7/site-packages (from nlt
k>=3.1->textblob) (1.0.1)
Requirement already satisfied: click in /Users/stamp/opt/anaconda3/lib/python3.7/site-packages (from nltk
>=3.1->textblob) (7.1.2)
Requirement already satisfied: regex in /Users/stamp/opt/anaconda3/lib/python3.7/site-packages (from nltk
>=3.1->textblob) (2020.11.13)
Note: you may need to restart the kernel to use updated packages.
```

In [23]:

```
from textblob import TextBlob
df['polarity']=df['review'].apply(lambda x:TextBlob(x).sentiment.polarity)
```

In [24]:

```
print("3 Random Reviews with Highest Polarity:")
for index,review in enumerate(df.iloc[df['polarity'].sort_values(ascending=False)[:3].index]['review']):
  print('Review {}:\n'.format(index+1),review)
```

```
3 Random Reviews with Highest Polarity:
Review 1:

D.T."
Review 2:
 "Best medicine I ever seen."
Review 3:
 "It has been excellent in curing my symptoms."
```

In [25]:

```
print("3 Random Reviews with Lowest Polarity:")
for index,review in enumerate(df.iloc[df['polarity'].sort_values(ascending=True)[:3].index]['review']):
  print('Review {}:\n'.format(index+1),review)
```

```
3 Random Reviews with Lowest Polarity:
Review 1:
 "I took this for restless leg syndrome and I had terrible involuntary muscle spasms and leg and arm mov
ement. I had to go to the ER. It was terrible."
Review 2:
 "So I&#039;ve had the implant since June 2015 and I haven&#039;t stopped bleeding it&#039;s so annoying
the reason I got it was so I wouldn&#039;t worry about getting a period every month but I&#039;ve been
on my period since I got it ! idk whether to leave it or just take it off ."
Review 3:
 "Drug works but the patch is horrible.  It is plastic and wrinkles up.  The edges don&#039;t adhere
well so the patch is always sticking to my clothes.  I hope they fix this issue.  If they do I will buy
again but for now I&#039;m going to save my money."
```

Down sizing Data set

In [26]:

```
df = df.sample(frac=0.05, random_state=1)
```

In [27]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10753 entries, 41735 to 150409
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   review    10753 non-null  object
 1   rating    10753 non-null  int64
 2   polarity  10753 non-null  float64
dtypes: float64(1), int64(1), object(1)
memory usage: 336.0+ KB
```

In [28]:

```
df.rating.unique()
```

Out[28]:

```
array([0, 1])
```

In [29]:

```
type(df.review)
```

Out[29]:

```
pandas.core.series.Series
```

In [30]:

```
import string
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

review_lines = list()
lines = df['review'].values.tolist()
stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()

def stem_words(text):
    return " ".join([stemmer.stem(word) for word in text.split()])

for line in lines:
    tokens = word_tokenize(line)
    # convert to lower case
    tokens = [w.lower() for w in tokens]
    # remove punctuation from each word
    table = str.maketrans('', '', string.punctuation)
    stripped = [w.translate(table) for w in tokens]
    # remove remaining tokens that are not alphabetic
    words = [word for word in stripped if word.isalpha()]
    # filter out stop words
#     stop_words = set(stopwords.words('english'))
#     words = [w for w in words if not w in stop_words]
    # Stemming
    # words = [stemmer.stem(w) for w in words]
    # Lemmatisation
#     words = [lemmatizer.lemmatize(w) for w in words]
    review_lines.append(words)

len(review_lines)
```

```
[nltk_data] Downloading package punkt to /Users/stamp/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /Users/stamp/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /Users/stamp/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[30]:

```
10753
```

In [31]:

```
from statistics import mean
print(mean([len(i) for i in review_lines]))
```

```
83.76341486096904
```

Develop a pre-trained embedding layer using Word2Vec

In [32]:

```
pip install --upgrade gensim
```

```
Requirement already satisfied: gensim in /Users/stamp/opt/anaconda3/lib/python3.7/site-packages (4.0.1)
Requirement already satisfied: numpy>=1.11.3 in /Users/stamp/opt/anaconda3/lib/python3.7/site-packages (f
rom gensim) (1.19.5)
Requirement already satisfied: scipy>=0.18.1 in /Users/stamp/opt/anaconda3/lib/python3.7/site-packages (f
rom gensim) (1.6.1)
Requirement already satisfied: smart-open>=1.8.1 in /Users/stamp/opt/anaconda3/lib/python3.7/site-package
s (from gensim) (5.0.0)
Note: you may need to restart the kernel to use updated packages.
```

In [33]:

```
import gensim

EMBEDDING_DIM = 100
# # train word2vec model
# # model = gensim.models.Word2Vec(sentences=review_lines, size=EMBEDDING_DIM, window=5, workers=4, min_
# model = gensim.models.KeyedVectors.load_word2vec_format('/Users/stamp/Desktop/Semester1-2021/IFN704/Go
model = gensim.models.Word2Vec.load('/Users/stamp/Desktop/Semester1-2021/IFN704/W2V_100/w2v_oa_all_100d.b
# # # vocab size
words = list(model.wv.index_to_key)
print('Vocabulary size: %d' % len(words))
```

```
/Users/stamp/opt/anaconda3/lib/python3.7/site-packages/gensim/similarities/__init__.py:15: UserWarning:
The gensim.similarities.levenshtein submodule is disabled, because the optional Levenshtein package
<https://pypi.org/project/python-Levenshtein/> is unavailable. Install Levenhstein (e.g. `pip install
python-Levenshtein`) to suppress this warning.
  warnings.warn(msg)
Vocabulary size: 3748342
```

In [34]:

```
model.wv.most_similar('sideeffect')#, topn =1)
```

Out[34]:

```
[('emphatic.', 0.8494895696640015),
 ('classes.most', 0.8362025618553162),
 ('effects)._because', 0.83547443151474),
 ('paresthesia”', 0.833727240562439),
 ('tomap', 0.8319109082221985),
 ('patients.simulation', 0.8307254910469055),
 ('pre-occupying', 0.8303970098495483),
 ('“headline”', 0.830180287361145),
 ('names.•', 0.8286046981811523),
 ('unchanging)', 0.8279871940612793)]
```

In [35]:

```
model.wv.most_similar('pain')#, topn =1)
```

Out[35]:

```
[('pain_and', 0.9086359739303589),
 ('pain_intensity', 0.8848637938499451),
 ('acute_pain', 0.8739394545555115),
 ('ongoing_pain', 0.8737969994544983),
 ('persistent_pain', 0.8737456798553467),
 ('pain,', 0.8705431818962097),
 ('pain_or', 0.8694646954536438),
 ('chronic_pain', 0.8577300906181335),
 ('intensity_of_pain', 0.8502391576766968),
 ('back_pain', 0.8502222895622253)]
```

In [36]:

```
model.wv.most_similar('cramp')#, topn =1)
```

Out[36]:

```
[('cramp,', 0.7645677924156189),
 ('cramp.', 0.7473682165145874),
 ('writer’s', 0.7147056460380554),
 ('muscle_pain', 0.6564078330993652),
 ('cramping', 0.6555342674255371),
 ('hemiplegic', 0.655203640460968),
 ('flat-arched', 0.6536697149276733),
 ('movement-evoked', 0.6526197195053101),
 ('spasticity_and', 0.6497231721878052),
 ('limping', 0.6473642587661743)]
```

In [37]:

```
# save model in ASCII (word2vec) format
# filename = 'drug_review_embedding_word2vec2.txt'
# model.wv.save_word2vec_format(filename, binary=False)
```

Start Building a classifier

In [38]:

```
import os

embeddings_index = {}
```

```
f = open(os.path.join('', 'drug_review_embedding_word2vec2.txt'),  encoding = "utf-8")
for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:])
    embeddings_index[word] = coefs
f.close()
```

In [39]:

```
from tensorflow.python.keras.preprocessing.text import Tokenizer
from tensorflow.python.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split

# VALIDATION_SPLIT = 0.3
max_length = 100

X_train, X_test, y_train, y_test = train_test_split(
    review_lines, df['rating'].values, test_size=0.3,
    shuffle = True, random_state=0)

# vectorize the text samples into a 2D integer tensor
tokenizer_obj = Tokenizer()
tokenizer_obj.fit_on_texts(X_train)
X_train_seq = tokenizer_obj.texts_to_sequences(X_train)
X_test_seq = tokenizer_obj.texts_to_sequences(X_test)

# pad sequences
X_train_pad = pad_sequences(X_train_seq, maxlen=max_length)
X_test_pad = pad_sequences(X_test_seq, maxlen=max_length)
word_index = tokenizer_obj.word_index
print('Found %s unique tokens.' % len(word_index))

# review_pad = pad_sequences(sequences, maxlen=max_length)
# sentiment =  df['rating'].values
# print('Shape of review tensor:', review_pad.shape)
# print('Shape of sentiment tensor:', sentiment.shape)

# split the data into a training set and a validation set
# indices = np.arange(review_pad.shape[0])
# np.random.shuffle(indices)
# review_pad = review_pad[indices]
# sentiment = sentiment[indices]
# num_validation_samples = int(VALIDATION_SPLIT * review_pad.shape[0])

# X_train_pad = review_pad[:-num_validation_samples]
# y_train = sentiment[:-num_validation_samples]
# X_test_pad = review_pad[-num_validation_samples:]
# y_test = sentiment[-num_validation_samples:]

# X_train_pad, X_test_pad, y_train, y_test = train_test_split(
#     review_pad, sentiment, test_size=0.2,
#     shuffle = True, random_state=42, stratify = sentiment)
```

```
Found 14725 unique tokens.
```

In [40]:

```
X_train_pad
```

Out[40]:

```
array([[   0,    0,    0, ...,  107,    8,   16],
       [ 614,    2,    6, ...,   93,  382, 8255],
       [   0,    0,    0, ...,  706,    4, 1141],
       ...,
       [   0,    0,    0, ...,  159,  264,  358],
       [   0,    0,    0, ...,  729,    7, 1931],
       [   0,    0,    0, ...,  130,   60,   90]], dtype=int32)
```

In [41]:

```
print('Shape of X_train_pad tensor:', X_train_pad.shape)
print('Shape of y_train tensor:', y_train.shape)

print('Shape of X_test_pad tensor:', X_test_pad.shape)
print('Shape of y_test tensor:', y_test.shape)
```

```
Shape of X_train_pad tensor: (7527, 100)
Shape of y_train tensor: (7527,)
Shape of X_test_pad tensor: (3226, 100)
Shape of y_test tensor: (3226,)
```

In [42]:

```
pip install imblearn
```

```
Requirement already satisfied: imblearn in /Users/stamp/opt/anaconda3/lib/python3.7/site-packages (0.0)
Requirement already satisfied: imbalanced-learn in /Users/stamp/opt/anaconda3/lib/python3.7/site-packages
(from imblearn) (0.8.0)
Requirement already satisfied: scipy>=0.19.1 in /Users/stamp/opt/anaconda3/lib/python3.7/site-packages (f
rom imbalanced-learn->imblearn) (1.6.1)
Requirement already satisfied: joblib>=0.11 in /Users/stamp/opt/anaconda3/lib/python3.7/site-packages (fr
om imbalanced-learn->imblearn) (1.0.1)
Requirement already satisfied: numpy>=1.13.3 in /Users/stamp/opt/anaconda3/lib/python3.7/site-packages (f
rom imbalanced-learn->imblearn) (1.19.5)
Requirement already satisfied: scikit-learn>=0.24 in /Users/stamp/opt/anaconda3/lib/python3.7/site-packag
es (from imbalanced-learn->imblearn) (0.24.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /Users/stamp/opt/anaconda3/lib/python3.7/site-pack
ages (from scikit-learn>=0.24->imbalanced-learn->imblearn) (2.1.0)
Note: you may need to restart the kernel to use updated packages.
```

Handle imbalanced classes with SMOTE

In [43]:

```
from imblearn.over_sampling import SMOTE

smote = SMOTE(sampling_strategy='minority')
X_train_sm, y_train_sm = smote.fit_resample(X_train_pad, y_train)
X_test_sm, y_test_sm = smote.fit_resample(X_test_pad, y_test)
```

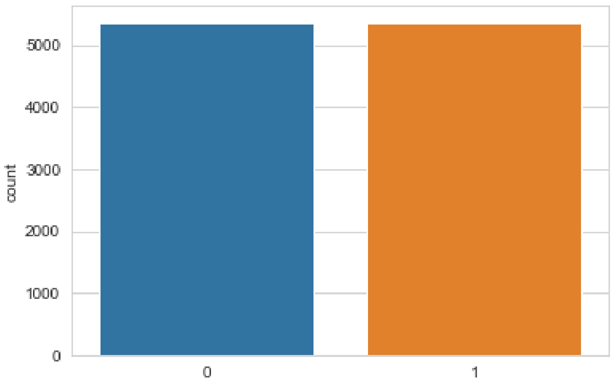
In [44]:

```
sns.set_style('whitegrid')
sns.countplot(y_train_sm)
```

```
/Users/stamp/opt/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass
the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will
be `data`, and passing other arguments without an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
```

Out[44]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f92e3722050>
```



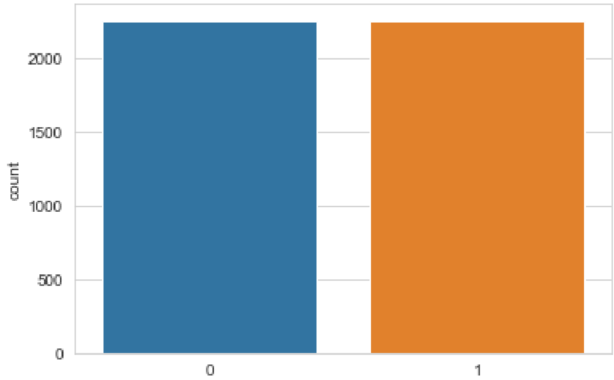
In [45]:

```
sns.set_style('whitegrid')
sns.countplot(y_test_sm)
```

```
/Users/stamp/opt/anaconda3/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass
the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will
be `data`, and passing other arguments without an explicit keyword will result in an error or
misinterpretation.
  FutureWarning
```

Out[45]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f92e36c1510>
```



In [46]:

```
print('Shape of X_train_pad tensor:', X_train_sm.shape)
print('Shape of y_train tensor:', y_train_sm.shape)

print('Shape of X_test_pad tensor:', X_test_sm.shape)
print('Shape of y_test tensor:', y_test_sm.shape)
```

```
Shape of X_train_pad tensor: (10716, 100)
Shape of y_train tensor: (10716,)
Shape of X_test_pad tensor: (4516, 100)
Shape of y_test tensor: (4516,)
```

In [47]:

```
from sklearn.model_selection import train_test_split
X_val, X_test_sm, y_val, y_test_sm = train_test_split(
                                    X_test_sm, y_test_sm, test_size=0.33, random_state=42, shuffle = True
```

In [48]:

```
EMBEDDING_DIM =100
num_words = len(word_index) + 1
embedding_matrix = np.zeros((num_words, EMBEDDING_DIM))

for word, i in word_index.items():
    if i > num_words:
        continue
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        # words not found in embedding index will be all-zeros.
        embedding_matrix[i] = embedding_vector
```

In [49]:

```
print(num_words)
```

```
14726
```

In [50]:

```
pip install keras
```

```
Requirement already satisfied: keras in /Users/stamp/opt/anaconda3/lib/python3.7/site-packages (2.4.3)
Requirement already satisfied: h5py in /Users/stamp/opt/anaconda3/lib/python3.7/site-packages (from keras
) (2.10.0)
Requirement already satisfied: pyyaml in /Users/stamp/opt/anaconda3/lib/python3.7/site-packages (from ker
as) (5.4.1)
Requirement already satisfied: scipy>=0.14 in /Users/stamp/opt/anaconda3/lib/python3.7/site-packages (fro
m keras) (1.6.1)
Requirement already satisfied: numpy>=1.9.1 in /Users/stamp/opt/anaconda3/lib/python3.7/site-packages (fr
om keras) (1.19.5)
Requirement already satisfied: six in /Users/stamp/opt/anaconda3/lib/python3.7/site-packages (from h5py->
keras) (1.15.0)
Note: you may need to restart the kernel to use updated packages.
```

In [51]:

```
from keras.models import Sequential
from keras.layers import Dense, Embedding, Flatten, LSTM, Dropout, BatchNormalization, GlobalMaxPooling1D
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D
from keras.initializers import Constant

# define model
model = Sequential()
# load pre-trained word embeddings into an Embedding layer
# note that we set trainable = False so as to keep the embeddings fixed
embedding_layer = Embedding(num_words,
                            EMBEDDING_DIM,
                            embeddings_initializer=Constant(embedding_matrix),
                            input_length=max_length,
                            trainable=False)

model.add(embedding_layer)
model.add(Conv1D(filters=256, kernel_size=6 ,padding='same', activation='relu', kernel_regularizer='l2'))
model.add(BatchNormalization())
model.add(SpatialDropout1D(0.5))
model.add(MaxPooling1D(pool_size=2))
# model.add(Dropout(0.25))
# model.add(BatchNormalization())
# model.add(Dropout(0.25))
model.add(LSTM(128, return_sequences=True))
model.add(LSTM(64))
model.add(Dropout(0.25))
model.add(Dense(32, activation=None))
# model.add(Dense(32, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(16, activation=None))
# model.add(BatchNormalization())
model.add(Dense(1, activation='sigmoid'))
print(model.summary())

# compile network
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# fit the model
history = model.fit(X_train_sm, y_train_sm, batch_size=64, epochs=15, validation_data=(X_val, y_val), ver
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 100, 100)          1472600
_________________________________________________________________
conv1d (Conv1D)              (None, 100, 256)          153856
_________________________________________________________________
batch_normalization (BatchNo (None, 100, 256)          1024
_________________________________________________________________
spatial_dropout1d (SpatialDr (None, 100, 256)          0
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 50, 256)           0
_________________________________________________________________
lstm (LSTM)                  (None, 50, 128)           197120
_________________________________________________________________
lstm_1 (LSTM)                (None, 64)                49408
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 32)                2080
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 16)                528
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 17
=================================================================
Total params: 1,876,633
Trainable params: 403,521
Non-trainable params: 1,473,112
_________________________________________________________________
None
Epoch 1/15
168/168 [==============================] - 72s 256ms/step - loss: 1.4507 - accuracy: 0.6310 - val_loss:
```

```
0.8852 - val_accuracy: 0.5336
Epoch 2/15
168/168 [==============================] - 40s 241ms/step - loss: 0.6707 - accuracy: 0.7540 - val_loss: 0.6536 - val_accuracy: 0.7382
Epoch 3/15
168/168 [==============================] - 41s 245ms/step - loss: 0.6190 - accuracy: 0.7830 - val_loss: 0.7984 - val_accuracy: 0.6350
Epoch 4/15
168/168 [==============================] - 41s 241ms/step - loss: 0.5811 - accuracy: 0.7931 - val_loss: 0.6301 - val_accuracy: 0.7696
Epoch 5/15
168/168 [==============================] - 41s 241ms/step - loss: 0.5763 - accuracy: 0.7896 - val_loss: 0.8513 - val_accuracy: 0.6013
Epoch 6/15
168/168 [==============================] - 41s 244ms/step - loss: 0.5539 - accuracy: 0.7975 - val_loss: 0.7372 - val_accuracy: 0.6988
Epoch 7/15
168/168 [==============================] - 41s 242ms/step - loss: 0.5518 - accuracy: 0.8031 - val_loss: 0.5799 - val_accuracy: 0.7805
Epoch 8/15
168/168 [==============================] - 43s 255ms/step - loss: 0.5198 - accuracy: 0.8201 - val_loss: 0.5105 - val_accuracy: 0.8162
Epoch 9/15
168/168 [==============================] - 41s 243ms/step - loss: 0.5003 - accuracy: 0.8318 - val_loss: 0.5475 - val_accuracy: 0.7983
Epoch 10/15
168/168 [==============================] - 43s 256ms/step - loss: 0.5021 - accuracy: 0.8248 - val_loss: 0.5788 - val_accuracy: 0.7550
Epoch 11/15
168/168 [==============================] - 893s 5s/step - loss: 0.5088 - accuracy: 0.8244 - val_loss: 0.8700 - val_accuracy: 0.6274
Epoch 12/15
168/168 [==============================] - 39s 235ms/step - loss: 0.5070 - accuracy: 0.8263 - val_loss: 0.5860 - val_accuracy: 0.7851
Epoch 13/15
168/168 [==============================] - 41s 244ms/step - loss: 0.4869 - accuracy: 0.8318 - val_loss: 0.6244 - val_accuracy: 0.7583
Epoch 14/15
168/168 [==============================] - 42s 252ms/step - loss: 0.4794 - accuracy: 0.8381 - val_loss: 0.5680 - val_accuracy: 0.8073
Epoch 15/15
168/168 [==============================] - 42s 251ms/step - loss: 0.4966 - accuracy: 0.8214 - val_loss: 0.4644 - val_accuracy: 0.8251
```

In [52]:

```python
from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())
```

```
[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 27074667297682800
]
```

In [53]:

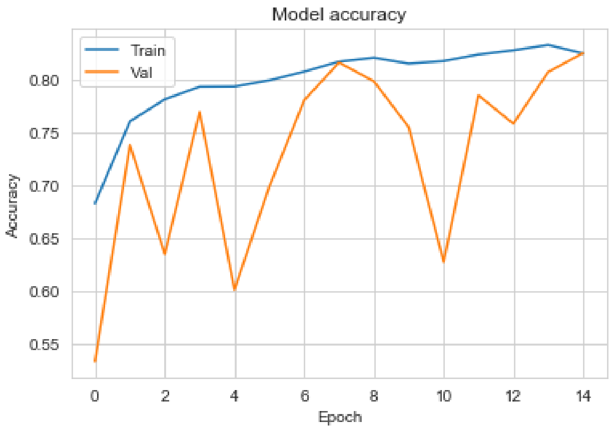
```python
import tensorflow as tf
print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
```

```
Num GPUs Available:  0
```

In [54]:

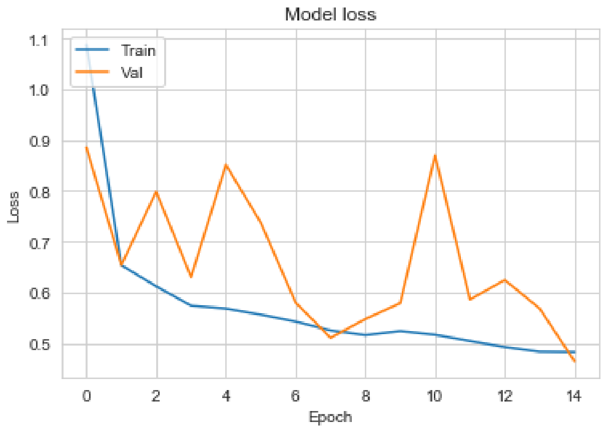
```python
import matplotlib.pyplot as plt
# %matplotlib notebook
%matplotlib inline

# Plot training & validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()
```



In [55]:

```
# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()
```



In [56]:

```
# evaluate the model
loss, accuracy = model.evaluate(X_test_sm, y_test_sm, batch_size=128)
print('Accuracy: %f' % (accuracy*100))
```

```
12/12 [==============================] - 2s 138ms/step - loss: 0.5029 - accuracy: 0.8028
Accuracy: 80.281693
```

In [57]:

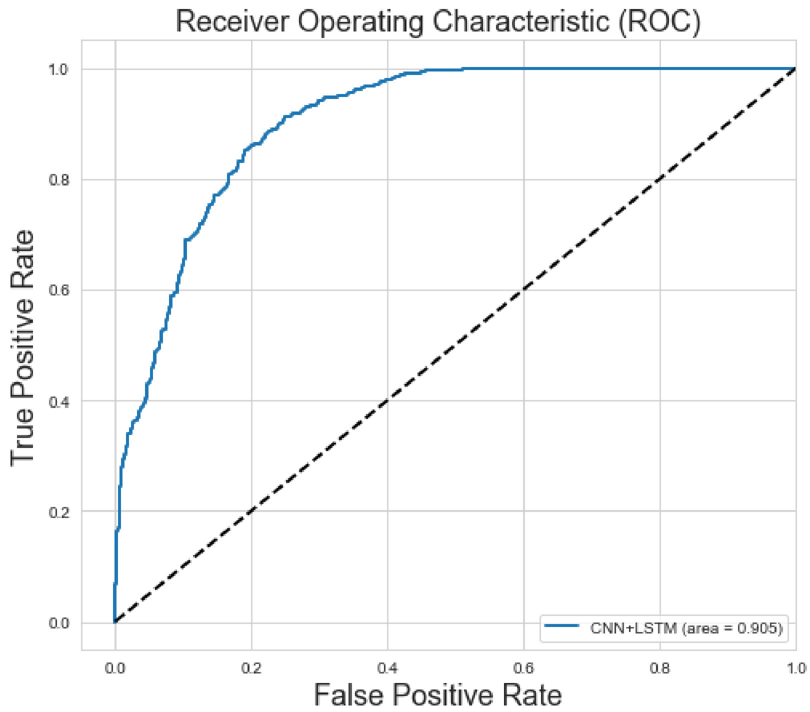
```
yhat = model.predict(X_test_sm)
```

In [58]:

```
from sklearn.metrics import roc_curve, auc
fpr, tpr, threshold = roc_curve(y_test_sm, yhat)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8,7))
plt.plot(fpr, tpr, label='CNN+LSTM (area = %0.3f)' % roc_auc, linewidth=2)

plt.plot([0, 1], [0, 1], 'k--', linewidth=2)
plt.xlim([-0.05, 1.0])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate', fontsize=18)
plt.ylabel('True Positive Rate', fontsize=18)
plt.title('Receiver Operating Characteristic (ROC)', fontsize=18)
plt.legend(loc="lower right")
plt.show()
```



More accuracy metrics

In [59]:

```
yhat_classes = model.predict_classes(X_test_sm)
```

```
/Users/stamp/opt/anaconda3/lib/python3.7/site-packages/tensorflow/python/keras/engine/sequential.py:450:
UserWarning: `model.predict_classes()` is deprecated and will be removed after 2021-01-01. Please use
instead:* `np.argmax(model.predict(x), axis=-1)`,   if your model does multi-class classification   (e.g.
if it uses a `softmax` last-layer activation).* `(model.predict(x) > 0.5).astype("int32")`,   if your mod
el does binary classification   (e.g. if it uses a `sigmoid` last-layer activation).
  warnings.warn('`model.predict_classes()` is deprecated and '
```

In [60]:

```
yhat_1D = yhat[:, 0]
yhat_classes_1D = yhat_classes[:, 0]
```

In [61]:

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
```

In [62]:

```
# accuracy: (tp + tn) / (p + n)
accuracy = accuracy_score(y_test_sm, yhat_classes_1D)
print('Accuracy: %f' % accuracy)
# precision tp / (tp + fp)
precision = precision_score(y_test_sm, yhat_classes_1D)
print('Precision: %f' % precision)
# recall: tp / (tp + fn)
recall = recall_score(y_test_sm, yhat_classes_1D)
print('Recall: %f' % recall)
# f1: 2 tp / (2 tp + fp + fn)
f1 = f1_score(y_test_sm, yhat_classes_1D)
print('F1 score: %f' % f1)
# ROC AUC
auc = roc_auc_score(y_test_sm, yhat_1D)
print('ROC AUC: %f' % auc)
# confusion matrix
matrix = confusion_matrix(y_test_sm, yhat_classes)
```

```
print(matrix)
```

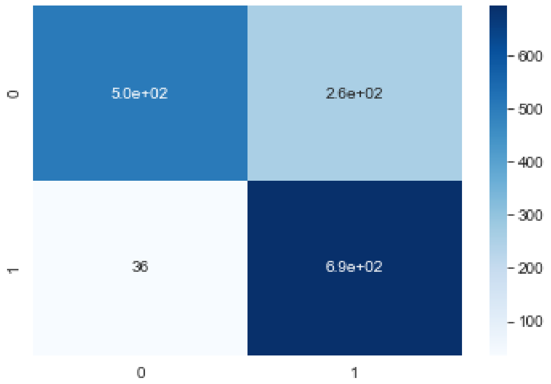
```
Accuracy: 0.802817
Precision: 0.728421
Recall: 0.950549
F1 score: 0.824791
ROC AUC: 0.905213
[[505 258]
 [ 36 692]]
```

In [63]:

```
sns.heatmap(matrix, annot=True, cmap='Blues')
```

Out[63]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f92e12c0bd0>
```



In [ ]: