Predicting Sentiment in Online Medication Reviews with Text-Based and Mixed-Data Models

Noah Selman
Raymond Eid
Angelo Cozzubo

Github: https://github.com/noahpselman/drug-review-nlp

Abstract

Our analysis extends previous research that aims to predict the sentiment of online medication reviews. We enhanced a public dataset used by previous researchers by applying web-scraping to more recent reviews. During preprocessing, we noticed 40% of examples used by previous researchers were duplicates. We trained a Logistic Regression, to compare with results from and LSTM model with both the duplicated and unduplicated, enhanced dataset to assess the impact of duplicates on model performance. We found little difference in accuracy for the Logistic Regression. However, the LSTM performed at a 10% higher accuracy on the dataset with duplicates. The implications of this result are unclear.

Next, we train two varieties of recurrent network-based models. The first are specifications of LSTMs that rely only on features derived from the text. The second are mixed data models that combine text-based features with metadata. The mixed data models perform 3-4% better. Improved preprocessing, time to experiment with other specifications, and more computing resources to facilitate training more epochs would have likely improved the accuracies we achieved. In the future we hope to extend this analysis in a myriad of ways.

Table of Contents

Introduction

Sentiment analysis can help identify useful texts from corposes too large to be efficiently interpreted by humans. In the domain of medicine and prescription drugs, such useful texts may include highly negative or positive patient experiences with certain prescription drugs. By applying an accurate sentiment analysis model to a corpus of medication reviews, medical professionals would have additional data to prognosticate patient reactions to drugs. For example, a doctor may find that individuals who had adverse reactions to drugs X and Y would reliably experience a specific side effect from drug Z. Such data would undoubtedly be very useful for medical practitioners prescribing medication and also to avoid adverse drug reactions.

We attempt to progress the field in this direction by building Recurrent Neural Network models that classify user reviews of medication found on drugs.com as positive, neutral, or negative.

Related Work

Our project is an offshoot of research by Grasser, Kallumadi, Malberg, and Zaunseder (2018), *Aspect-Based Sentiment Analysis of Drug Reviews Applying Cross-Domain and Cross-Data Learning*.

The authors aimed to predict the categorical sentiment of data they gathered 215,063 medication reviews by scraping two websites on which users post reviews of their personal experiences with the drugs.[1] In their analysis, the authors trained a Logistic Regression model that predicted the correct label on 92% of examples.

The authors note that the model performs worse on examples labeled as Neutral, likely due to the fact that this class is underrepresented in the dataset. The authors attempted to compensate for this disproportionate distribution in model training by penalizing classification errors with a weight inversely proportional to class frequency. The authors conclude by noting the varied accuracy of their model in predicting consumer reviews across consumer conditions and emphasizing the need for more sophisticated feature design and larger datasets.[2]

In another study using the same dataset with all 215,063 rows, authors Colon-Ruiz and Segura-Bedmar (2020) apply deep learning models to classify drug reviews.[3] The authors explore the efficacy of neural networks in the same sentiment analysis task, and also attempt a finer grained (10 class) polarity classification of drug reviews.

---

[1] More details in the following section.
[2] The seminal paper also uses data collected on druglib.com and compares performance of models across data sources. Our analysis focuses exclusively on the components of the authors' analysis dealing with just the drugs.com data.
[3] Colon-Ruiz, Segura Bedmar, *Comparing deep learning architectures for sentiment analysis on drug reviews* (2020)

The authors use a variety of word embeddings that use sources such as PubMed, Wikipedia, and informal posts from Twitter about drugs and a range of different deep learning architectures. These architectures include a simple CNN, a bidirectional LSTM, a hybrid model using a CNN concatenated with a bidirectional LSTM, and a hybrid model consisting of a bidirectional LSTM followed by a CNN. Their fifth and final model is a BERT with an LSTM as a classifier, which presents a distinct advantage over other frameworks as the model takes words in their contexts, so different meanings for the same word are represented. Although the architecture of the final model enhanced performance, it came at a significant computational cost.

Their results in the 10-class classification task varied markedly across models. The CNN with pre-trained static embeddings perform poorly due to its inability to extract information from long-range semantic dependencies. The hybrid models (LSTM + CNN) do not truly outperform the simpler LSTM models. As expected, predicting 10 classes is a more difficult task than the 3-class classification given that reviewers subjectively assign rating.

Using a three-class classification, the authors found their best model (the BERT + LSTM) falls short of the Grasser et al.'s Logistic Regression in accuracy score, 0.9046 versus 0.9224. Neural networks tended to accurately predict strong sentiments, with the BERT + LTSM attaining an F1 score of 0.9477 for positive polarity and 0.872 for negative. Neutral was the most difficult category to predict, with the highest F1 score of all models attained by the BERT + LSTM, at only 0.65. Like the authors of the seminal paper, Colon-Ruiz and Segura-Bedmar suspect this to be the case because the class represents only 10% of all instances.

Other papers have been written using Grasser et al's dataset, including Mo and Biseda (2020)[4], who used 8 different BERT models for this sentiment analysis task. These models were variations of BioBERT, a pre-trained biomedical word representation model, and Clinical BERT, a pre-trained model created from clinical text and discharge summaries. Using a max sequence length of 128, a training batch size of 32, and a learning rate of $2e^{-5}$ as their hyperparameters over 4 epochs, the best accuracy attained was 88.9% with the Clinical Bert Discharge model. Despite the complexity and nuance of Neural Network models, Grasser et al's logistic regression accuracy has yet to be surpassed.

Data

Our data source is drugs.com, a website which provides information on an immense array of medication and where users can share reviews of drugs they have used or are currently using. Reviews vary in length up to 300 words and are often associated with a 1-10 rating. We attempt to use the text of the review to predict a sentiment, given by binning ratings.[5] The content of the reviews is informal, containing misspellings, slang, references to experiences with other drugs, and colorful capitalization patterns, among other informalities. Some of these characteristics

[4] Mo and Biseda, *Enhancing Pharmacovigilance with Drug Reviews and Social Media*
[5] Details in methodology section

make traditional methods for sentiment analysis difficult to apply. The following positive review for an obesity medication illustrates this example.

"Medication started May 1st. I take 30mg in the morning around 7:30. First couple nights I couldn't sleep and I had headaches but both subsided. I have changed everything. I'm 5'9 333lbs to start. Today I weighed in at 317. I decided while taking this to get a grip on snacking and eating etc. I'm eating healthier still the same stuff I would normally eat but now I'm eating 1300 calories a day with the help of my fitness pal. I'm also exercising 30-45 minutes a day. I rotate workouts, I'll do aerobics, use the treadmill, ride a bike, or go swimming. Swimming laps is my favorite because I feel less tired after. But even if I use the treadmill at night 9pm, when I get done I still have so much energy I tend to dance to to continue my cool down"

While the user assigned a positive rating, even a human might have difficulty identifying this due to meandering language and an absence of declarative statements. This suggests that a model using long-range memory over a review (RNNs) will be more effective than those using local dependencies (CNN). As will be discussed in the methodology section, we apply various implementations of Long Short Term Memory (LSTM) recurrent neural networks in an attempt to capture information throughout the entire review.

The foundation of our dataset is the same used by Grasser et al in the 2018 paper. Below we provide a summary of the data.

Source Link: https://archive.ics.uci.edu/ml/datasets/Drug+Review+Dataset+(Drugs.com)
Data Date Range: January 1, 2008 to January 31, 2017
Features (data type):

- drugName (categorical): name of drug
- condition (categorical): name of condition
- review (text): patient review
- rating (numerical): 10 star patient rating
- date (date): date of review entry
- usefulCount (numerical): number of users who found review useful

Number of examples: 215,063 from Drugs.com

In our investigation of the dataset we found a nontrivial number of duplicated drug reviews. As shown in Figure 1 below, the same review and rating appear for two different drugs. Upon inspection of their description, we find that grouped drugs may be nearly identical and in this case, Quetiapine is simply the generic version of Seroquel. While we did not investigate whether drugs.com automatically populates one user review among all related drugs, this may be a worthwhile endeavor, and could have serious implications for the validity of specific drug reviews

Figure 1. An example repeated review

drugName	condition	review	rating	date	usefulCount
Seroquel	Schizoaffective Disorde	"No problems, watch what you eat."	10.0	October 8, 2014	19
Quetiapine	Schizoaffective Disorde	"No problems, watch what you eat."	10.0	October 8, 2014	19

Medication descriptions from drugs.com:

Seroquel (quetiapine) is an antipsychotic medicine. It works by changing the actions of chemicals in the brain. Seroquel is used to treat schizophrenia in adults and children who are at least 13 years old.
Quetiapine is an antipsychotic medicine that is used to treat schizophrenia in adults and children who are at least 13 years old. Quetiapine is used to treat bipolar disorder (manic depression) in adults and children who are at least 10 years old.

Compiled by authors

In total, 86,585 rows of data had identical reviews and ratings in the original dataset, roughly 40% of all reviews, and cleaning the data resulted in 128,478 unique review-rating pairs. This prompted us to question the accuracy of models in previous research studies, and to explore if the reported results were overestimated. We evaluate various models using the original duplicated dataset and cleaned datasets in the Components of Analysis section of this paper.

In addition to dropping rows containing duplicated reviews, we augmented the dataset by scraping more recent data from drugs.com. The original dataset that was scraped and made available by Grasser et al. ranged from January 2008 to January 2017. We extended it by scraping data from February 2017 to May 4, 2021 using Python's Beautiful Soup library. After dropping duplicates, we obtained 56,520 new drug reviews and concatenated them to the original, non-duplicated dataset, resulting in a total of 184,998 rows. In this enhanced dataset, the number of rows in the *Positive* rating category is 115,426, or 62.4% of all data, rows in *Negative* is 48,121, or 26% of rows, and the least represented being *Neutral*, at 21,451 rows and 11.6% of the data.

Components of Analysis

Our research has three related but separate components.

1. Exploring the implications of duplicated drug reviews in the original dataset on previous research
2. Creating a model that uses only text-based features to predict sentiment
3. Creating a model that uses text-based features in addition to metadata of the dataset

In preparation for training the models, all 3 methodologies bin the rating into the following categories.[6]

- Positive: 7-10
- Neutral: 4-6
- Negative: 1-3

Additional pre-processing details along with specifics on the methodology, results, discussion, successes, failures, and future work will be discussed in the following sections.

Each model uses Cohen's Kappa to identify the optimal set of parameters after each epoch. Cohen's Kappa is used in lieu of accuracy because it accounts for the possibility that labels were correctly predicted by random chance. This method is consistent with the literature.

The effect of duplicates on previous research

Discussion of leakage

Although the training and test examples are independently and identically distributed (iid) and technically do not constitute leakage, previous research models are experiencing bias akin to that caused by leakage. This leakage-like bias is due to the presence of repeated rows shared by both the test and training set. In this section we explore the effect of dropping duplicates on accuracy and learning.

As illustrated in Figure 2, a correctly performed split of a dataset into train and test sets creates two distinct and disjoint sets that are independently and identically distributed samples from the superset. An incorrect sampling that violates the iid assumption would lead to duplicated rows appearing in both the testing and training sets, as depicted in Figure 3. Lastly, Figure 4 demonstrates how bias occurs in the original dataset despite no violation of the iid assumption in the train-test split.

[6] Note that this is slightly different from the binning rule used by previous papers. Other authors categorized reviews with a rating of 4 as negative. Based on personal experience, we decided to categorize examples with a rating of 4 a neutral instead. This may have had a negative effect on our models' accuracies, as they proved worse at correctly predicting neutral-labeled examples.

Figure 2: A correct train-test split of the data

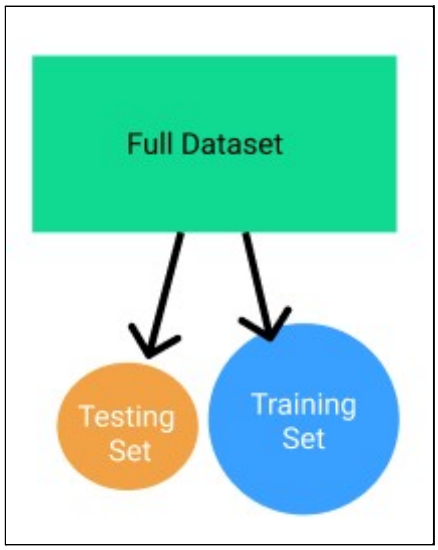


Figure 3: A train-test split with leakage

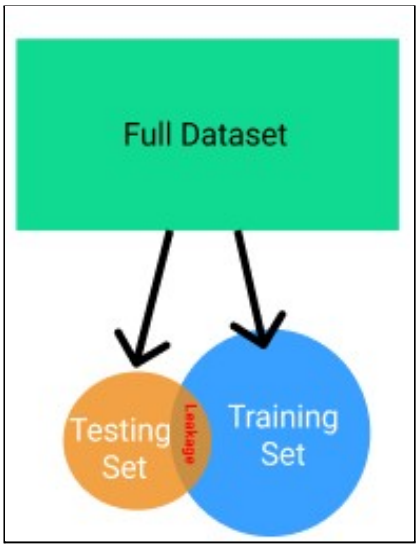
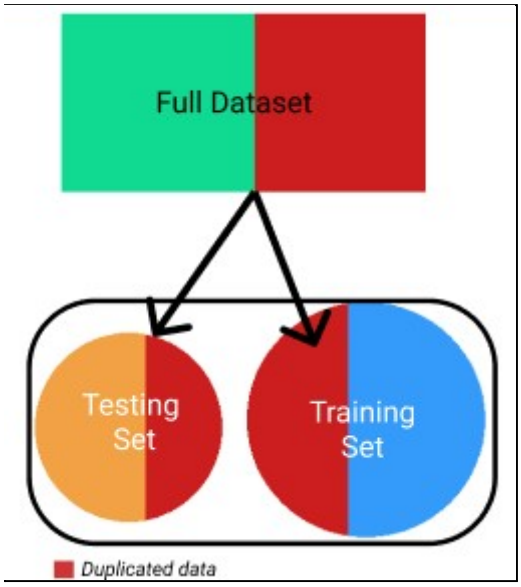


Figure 4: A correct split that behaves like leakage



Compiled by authors

The bias introduced by duplicated reviews results in the test set containing 32,216 rows (60%) also present in the training set. As Kaufman, Rosset & Perlich [7] state in their 2011 paper, leakage is a source of poor generalization and overestimation of expected performance. The following section explores the degree to which previous researchers overestimated their model's performance.

[7] Kaufman, Rosset & Perlich, *Leakage in data mining* (2011)

Re-evaluating the models with cleaned datasets:

Given the duplicates in the dataset, we undertook a comparison of two models trained with two datasets: the original dataset and datasets containing unique rows.
First we implement a Logistic Regression to investigate the effects of duplicates on the Grasser paper. Next, we train an LSTM[8] to investigate the effects of duplicates in the Colon-Ruiz analysis. We found that the logit's results did not change significantly across datasets whereas there was a marked difference with the LSTM.[9]

In Figures 5 and 6 below, we see the results of two Logistic Regression models. The first was trained on the original dataset while the second was trained on the original dataset with no duplicates. Both regressions were run over 200 epochs using the Adam optimizer with an initial learning rate of 0.001, and Cross Entropy Loss. The first model attained a test set accuracy of 0.780, while the second model attained an accuracy of 0.775. This small difference in accuracy was unexpected. We expected the presence of duplicated rows to create a greater discrepancy between accuracy of the two models.

We attempted but were not able to complete a full training of a third Logistic Regression model trained on the non-duplicated and enhanced dataset due to resource constraints. Nonetheless, in 170 epochs of training, the third model attained a test set accuracy of 0.76. The result of the third model was also surprising. We expected a greater number of observations leading to a higher accuracy than the second model. While we fell short of the 0.92 accuracy score of Grasser et al.'s logit, our three models suggest that if the authors were to re-run their code with a non-duplicated dataset, their accuracy would not be significantly diminished[10]. We postulate that this is the case because a Logistic Regression is not a very complex model - consisting of one single linear layer - and its parameters are not fitting too closely to noise in the data.

The LSTM displays a stronger sensitivity to duplicates. Both models in Figures 7 and 8 were trained over 30 epochs, the Adam optimizer with an initial learning rate of 0.001, and Cross Entropy Loss. The first model was trained using the original dataset and attained a test set accuracy of 0.90 and the second model was trained on the non-duplicated enhanced dataset and attained a test set accuracy of 0.80. These results fell more closely in line with our expectations as the gap is 0.1. The first model is performing very well due to the data "leakage". Although we did not train a third model with the original, non-duplicated dataset, we expect that the model's accuracy would be even lower than that of the second model as it would be trained over fewer rows. We postulate that this large difference in accuracy between these two models arises because an LSTM has a complex functional form with a vastly greater number of parameters compared to the Logistic Regressions. These parameters allow the model to better learn the training examples that "leak" into the test set. An LSTM model trained with data leakage will attain a more artificially high test set accuracy than a simpler Logistic Regression.

[8] LSTM specification details provided later.
[9] Unfortunately, a few short-comings in pre-processing harmed the integrity of this analysis.
[10] This came entirely as a surprise and we carefully vetted our code to verify our results.

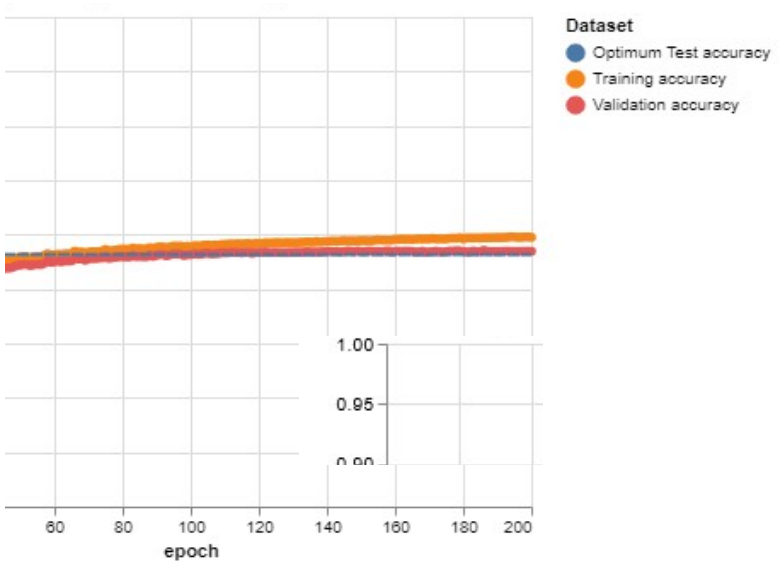


Figure 5: Logit trained on original duplicated dataset

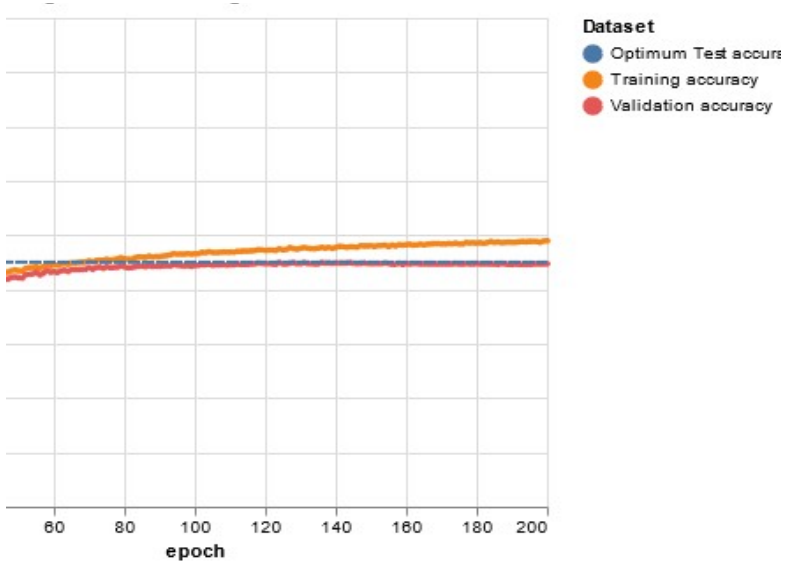


Figure 6: Logit trained on original non-duplicated dataset

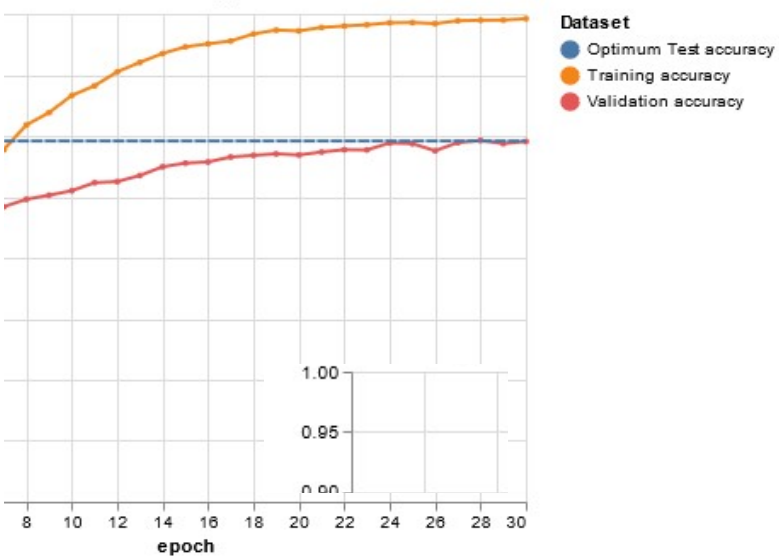


Figure 7: LSTM trained on original duplicated dataset

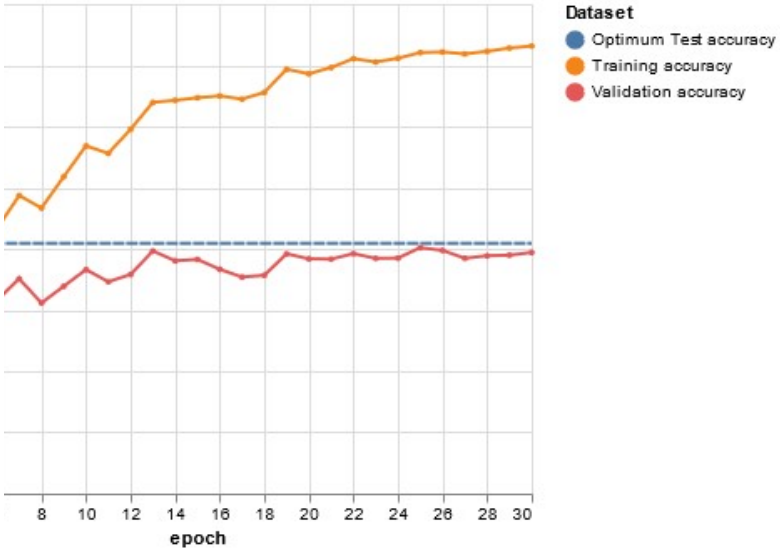


Figure 8: LSTM trained on enhanced, non-duplicated dataset

Models using only Text-based Features

This section compares models that exclusively use text-based features to predict sentiment in various specifications of the LSTM model. The advantage of limiting these models to text-based features allows the resulting algorithm to be applied to arbitrary drug reviews, with or without meta-data.

Pre-processing

These models were implemented with the PyTorch Python package using the Spacy tokenizer. Below are the pre-processing steps used to prepare data the data:[11]

1. We removed examples with no rating. Because rating is our target, we cannot use examples without that information.
2. We then dropped duplicates based on the review and date column.
3. We replaced drug names found in the review with the token 'THISDRUG'. This allowed the recurrent network to differentiate between the drug that is the subject of the review (if spelled correctly) and other drugs that are mentioned as comparisons.
4. We executed two data cleaning steps: stripping quotations wrapping the reviews and replacing HTML symbols with the corresponding punctuation.[12]
5. We binned our rating according to the rules described above.
6. Then, we tokenized the review column. Because the spacy tokenizer creates numerous useful fields from text, we saved a few pieces of information as lists with elements corresponding to each token. These pieces of information include parts of speech, shape of the word (e.g., lengths and capitalizations), dependencies, and lemmas. An example from the part of speech column might look as follows: '[NOUN, VERB, PREPOSITION]'.
7. Finally, to facilitate one-hot encoding generation that would happen as our data is loaded, we converted the aforementioned lists of tokens into lists of indices, wherein each index represents a one-hot encoding for that word. An example for part of speech might look as follows: '[3, 6, 1]' where each number represents an index in a vocabulary of parts of speech.

There is room for improvement in our pre-processing pipeline. There are three steps we would have liked to take if we had more time and resources:

1. We realized our string cleaning was imperfect at a point too late to correct. Many strings included incorrectly decoded HTML symbols. This could have materially hurt our models

[11] As noted above, some of the data was downloaded from the Grasser paper, while the rest was obtained in a scrape of drugs.com.
[12] A later analysis revealed that our cleaning was inadequate. Our rules failed to remove a large portion of the html which may have meaningfully impacted our results. More details on this later.

performance, as these symbols would have been embedded as unknowns as opposed to the correct, pre-trained vector.
2. Moreover, it would have been useful to experiment with some spell-correcting algorithm. This may have cleaned up some of the typos, allowing those tokens to be appropriately embedded. Unfortunately, resource constraints limited our scope to just part-of-speech and dependency data.
3. We would have liked to have built a vocabulary of all the drug names and replaced drugs that were not the subject of the review with a special symbol, (perhaps OTHERDRUG). Hopefully models would learn to not pay attention to descriptors of that token.

Model Specifications

All used text features in these models are word-specific. As such, we concatenated the text feature to the corresponding word embeddings and passed through the recurrent neural network.

All specifications we ran share the same general structure. They start by transforming words to tre-trained, unfrozen glove embeddings with 300 dimensions. The embeddings are passed through a bi-directional LSTM network with two layers and 50% dropout. The recurrent layer feeds its output to a 300 dimension linear layer that maps to a 3 dimensional output corresponding to our labels. Additionally, there is a 50% dropout after the embedding and LSTM layer.

All specifications also use a weighted cross entropy loss. The weights for each class are the inverse of the classes' proportion in the training dataset. Due to dataset imbalance, this forces the model to pay extra attention to patterns in the neutral classes as opposed to just the positives.

All specifications also use the Adam Optimizer algorithm which allows us to have an adaptive learning rate for each parameter. The learning rate is initialized to 0.001.

For each specification, we evaluated the best model on our test dataset. During training, the model was evaluated on the validation set after every epoch. The best model was defined as the model with the highest Cohen's Kappa. We used Cohen's Kappa (as opposed to accuracy) because it considers correct classifications that may have resulted from random chance. Each model was trained for 30 epochs.

We train three models of the following three specifications:

- Model 1: No additions to the specifications above. This model is run on the full dataset and de-duplicated Grasser enhanced with scraped data.
- Model 2: The above specifications but using a lemmatized token as opposed to the text tokens.

- Model 3: The base specifications with part-of-speech and dependency one-hot encodings concatenated to each word embedding prior to being passed through the recurrent layer

Results

Interestingly, neither of the latter two specifications improved the accuracy of the model. We suspect that the shortcomings in pre-processing[13] prevented these additional components from extracting relevant information from the tokens. Table 1 describes the performance of each model using two evaluation metrics: accuracy and Cohen's Kappa. In the following section, we present detailed results and discussion for each specification.

Table 1. LSTM performance

Model (dataset)	**Accuracy**	**Cohen's Kappa**
Model 1 (full)	0.805	0.647
Model 1 (Grasser)	0.896	0.788
Model 2 (full)	0.800	0.639
Model 3 (full)	0.804	0.646

Base Model

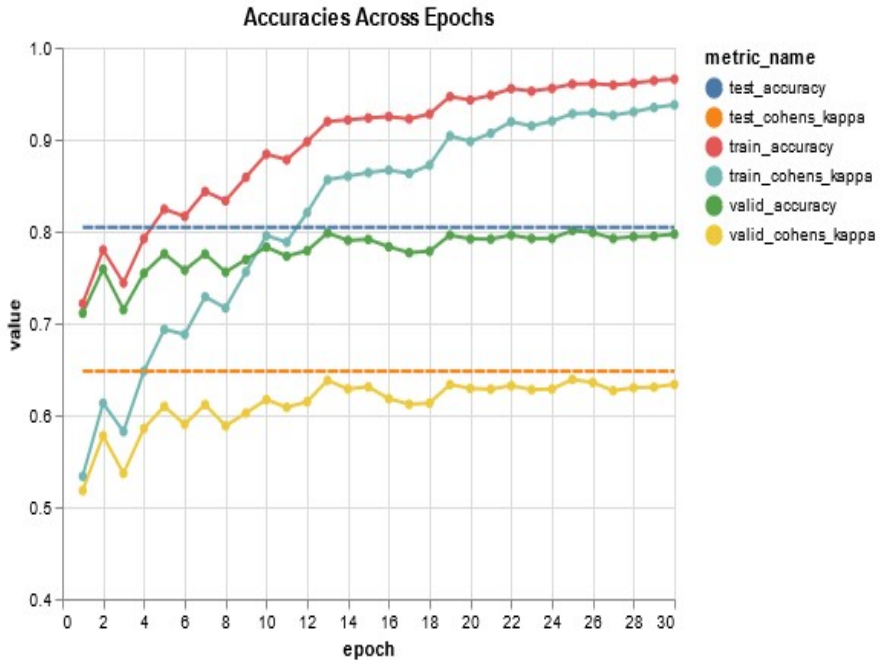
Evaluated on the full data set, our base model had an accuracy of 0.805 and a Cohen's Kappa of 0.647. Figure 9 shows the accuracy and Cohen's Kappa for the train, validation, and test datasets in each epoch.

As the figure depicts, the validation accuracy essentially plateaued around the 13th epoch. Training accuracy was still increasing at this point and it's possible there may have been additional gains if we were able to continue training.

Figure 10 displays the accuracy for each class for the validation dataset over the epochs.

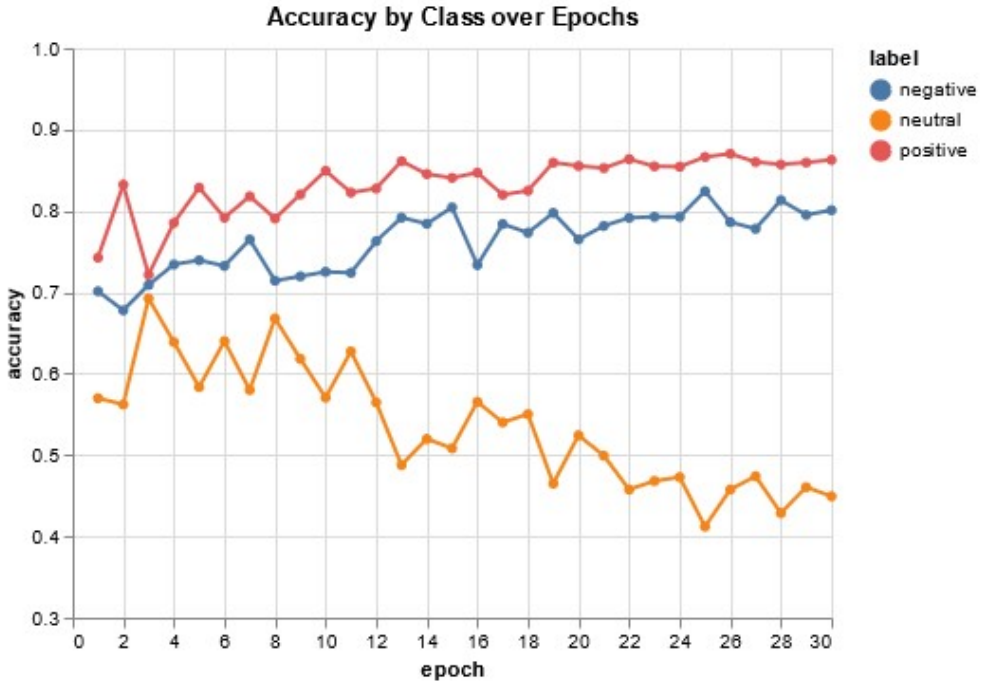
[13] Particularly an incomplete removal of nonsensical symbols

Figure 9



Compiled by authors

Figure 10

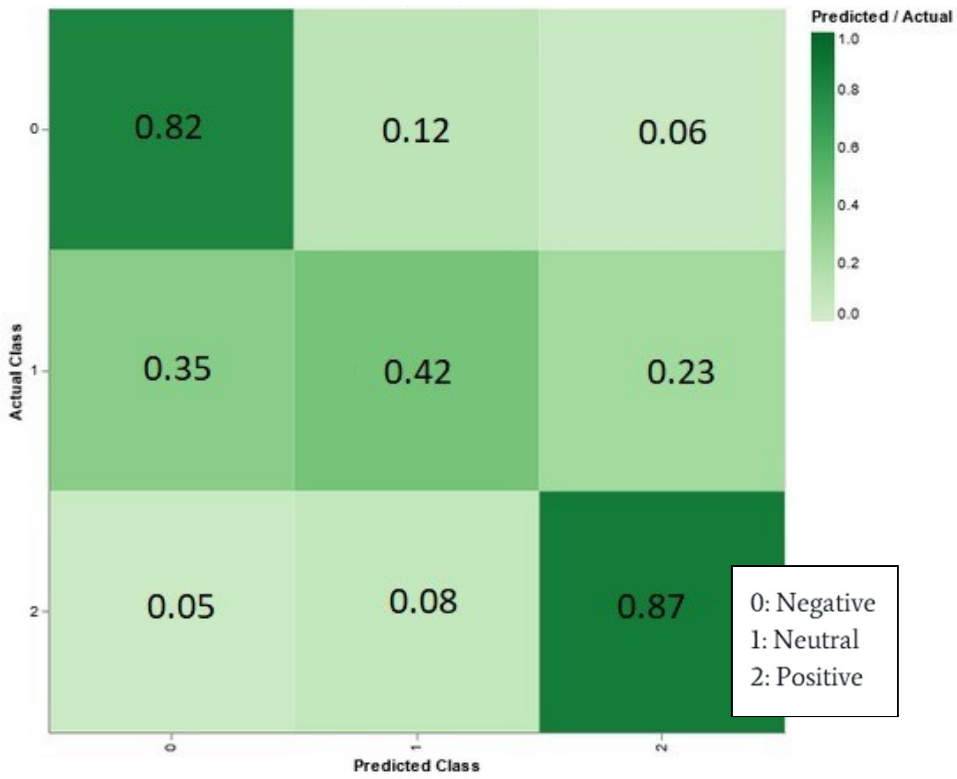


Compiled by authors

Each line represents the proportion of each correctly labeled class example over all true examples of that classification. This chart shows that while classification accuracy for positive and negative examples improves gradually throughout training, classification accuracy for neutral examples degrades. We attempted to address this problem by using a weighted loss function (described earlier). Nonetheless, this chart suggests that the patterns that the model learned which correctly classify some positive and negative examples also caused some neutral examples to be misclassified. For example, at epoch 25, negative classification improved while neutral classification decreased.

This suggests the model learned a pattern that existed in both neutral and negative examples and classified examples exhibiting that pattern as negative. This hypothesis is supported by anecdotal browsing of the dataset. Many reviews labeled as neutral are difficult to distinguish from positive and negative reviews. Throughout training there seems to be an inverse relationship between change in positive and negative accuracy and change in neutral accuracy. The confusion matrix in Figure 11 further supports this point.

Figure 11



Compiled by authors

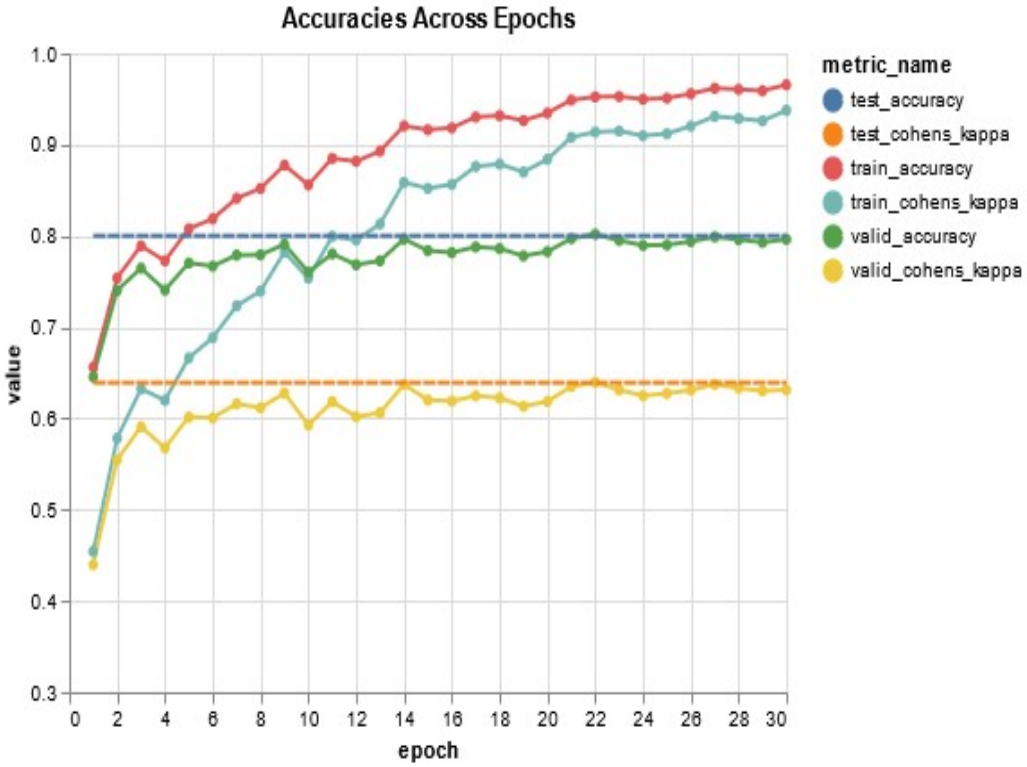
The number in each cell represents the proportion of examples in the y-axis predicted as the x-axis. As you can see, the model more often mis-classifies examples labeled positive and negative (2 and 0) as neutral than as the opposite class. Neutral-labeled examples are correctly

labeled more often than random chance but not significantly so. They're more frequently misclassified as negative than positive.

Lemmatized Model

We produced the same charts to analyze the performance of our lemmatized model. The results do not differ greatly thus the discussion will be brief.

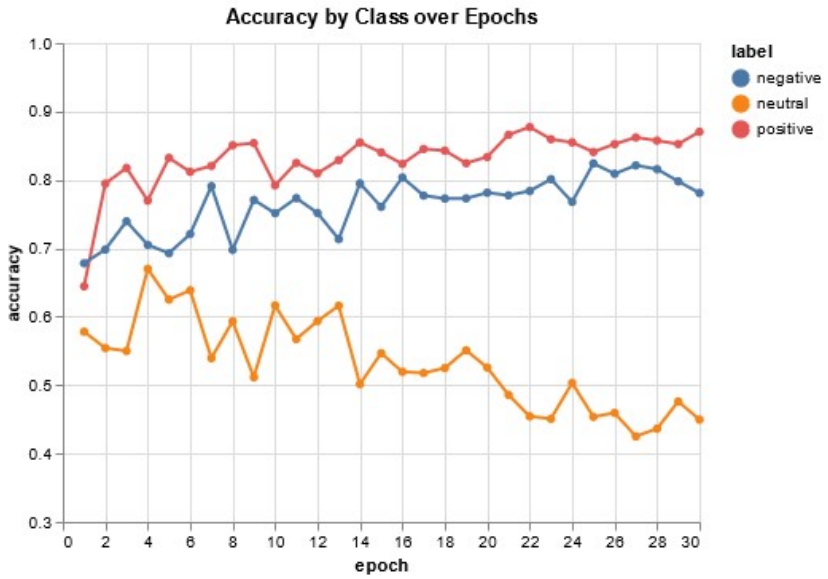
Figure 12



Compiled by authors

In Figure 12 the validation gain tapers off around epoch 13 as well. Performance on the test dataset is very similar to the base model.

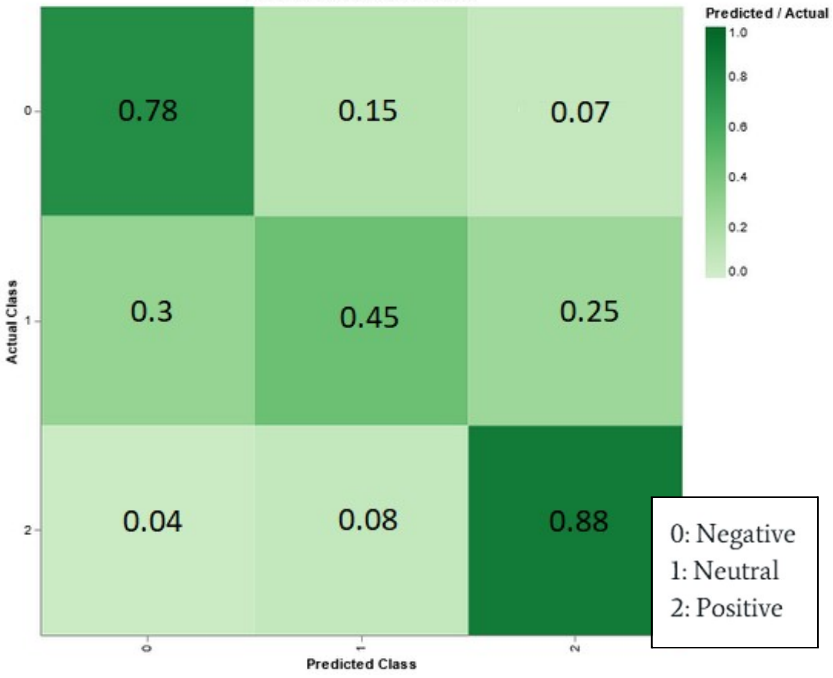
Figure 13



Compiled by authors

Similar to the base model, positive and negative accuracy increase gradually while neutral accuracy slowly declines. The aforementioned inverse relationship between the positive and negative accuracy change and the neutral accuracy change exist here as well.

Figure 14



Compiled by authors

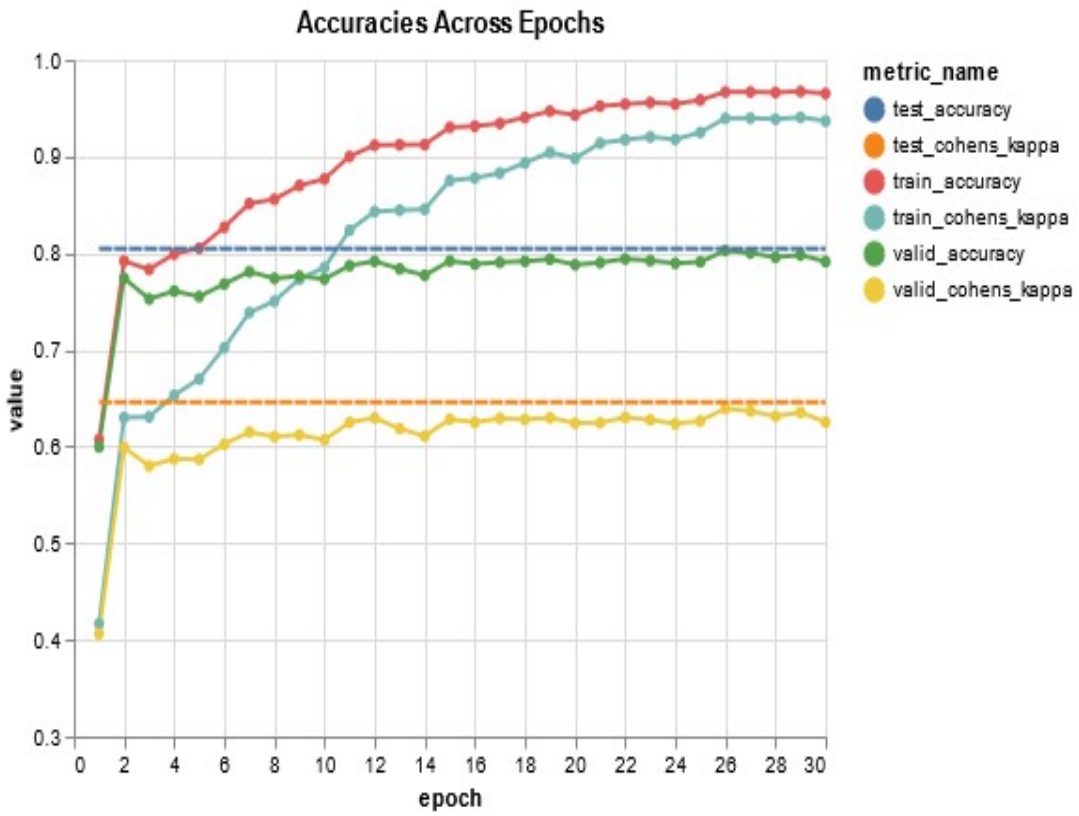
The confusion matrix reveals similar patterns as with the base model.

While it is surprising that the model doesn't improve when using lemmatized tokens, we provide a few potential explanations. First, aforementioned shortcomings in our pre-processing may have prevented the gains that lemmatization would be expected to bring. More precisely, tokens with uncleaned symbols cannot be lemmatized. Secondly, the Glove embeddings are trained on un-lemmatized text. By lemmatizing our inputs, we may have been removing relevant information stored in the specific manifestations of the lemmas.

Part-of-Speech and Dependency Features

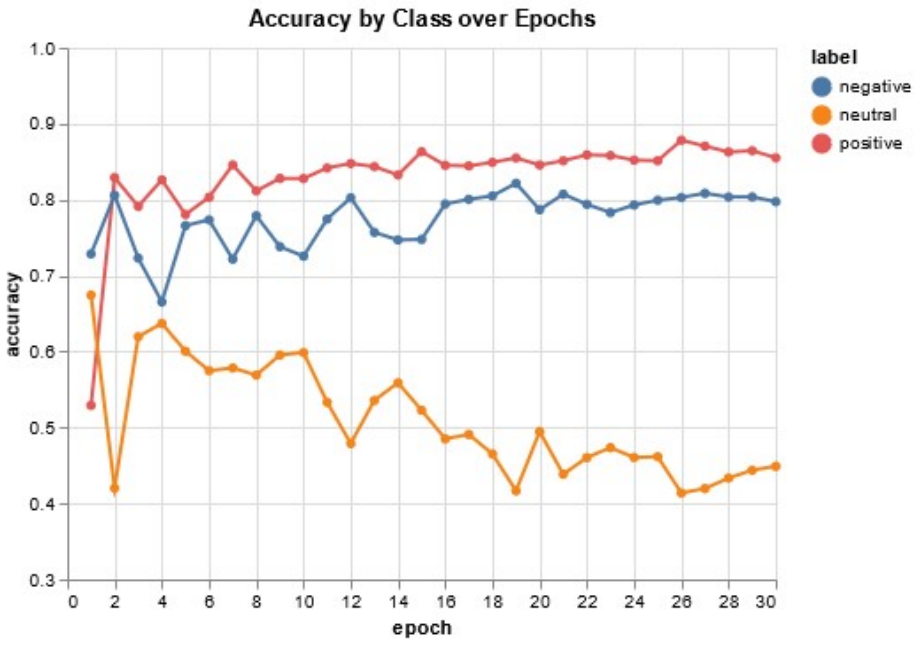
As with the lemmatized model, the analysis of the model with part-of-speech and dependency features shows patterns similar to those of the base model. Therefore discussion will be brief. For this specification, validation accuracies and Cohen Kappa's appear to taper off even earlier than in previous models. Figure 16 brings more light to this phenomenon.

Figure 15



Compiled by authors

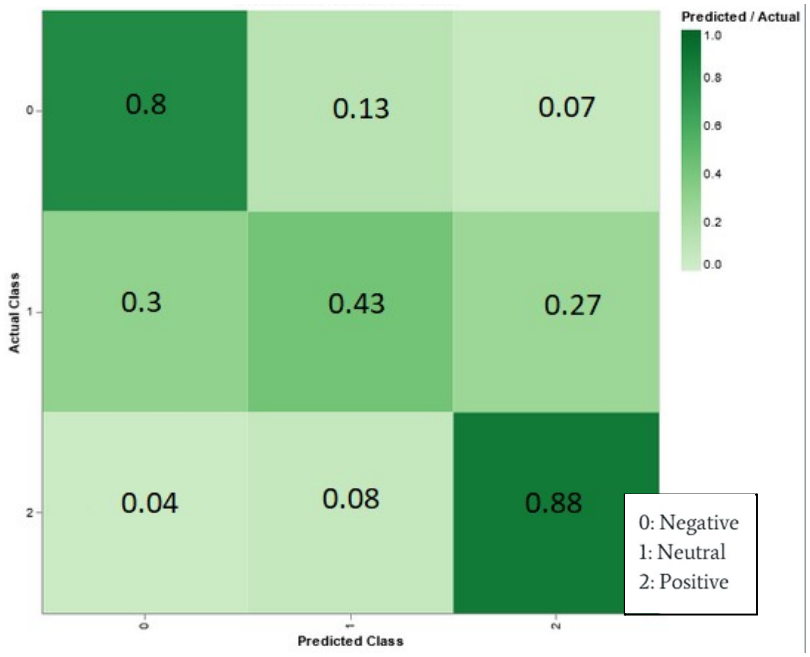
Figure 16



Compiled by authors

For this run, the model started predicting, perhaps by random coincidence, neutral-labeled examples better than positive. Neutral accuracy declined from then on and appears to have balanced the gradual gains of predicting the other labels correctly.

Figure 17



Compiled by authors

The confusion matrix shows similar patterns to the previous ones.

Future Work

Limited time and computing resources prevented us from running further specifications as well as the same specifications with improved pre-processing. A few examples of models we'd like to run are as follows

- The base specification but with part-of-speech and dependency one-hot encodings, and name-entity recognition, word shape data encodings. This might present computational challenges, as there are many different word shapes and entities appearing in the data. Therefore, any one-hot vector could be quite long. However, this may provide the recurrent layer additional relevant information and thus improve model performance.
- The base model but with an additional medical specific embedding concatenated to glove embedding.[14] Because the additional embedding would provide domain-specific context, it may add valuable information to the LSTM layer.
- We could have tried other popular models for long-distance attention in natural language processing such as a bi-directional encoder representation transformer.

Models that Include Metadata

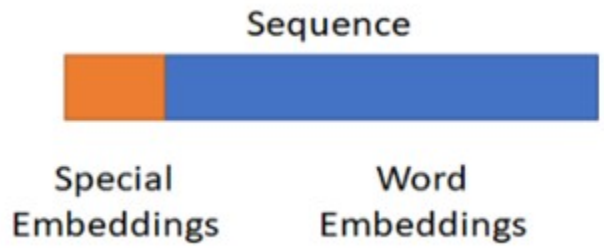
Here we consider the use of metadata features to provide the model additional information. The literature calls models that combine different types of features mixed data models. Previous research in this area by Grasser et al. (2018) and Colon-Ruiz and Bedmar (2020) has not exploited these additional features. Hence, exploiting these variables would be a natural extension of our models. Additional features were available in both the data we obtained from Grasser and the data we directly scraped.

Including metadata required us to extend our models to include numeric variables in a tabular form. Benefits to this model included the possibility of discovering complementary predictive patterns not found in the text data. Furthermore, we believe the model we trained is applicable in real-life situations because the metadata used will nearly always be available. Using mixed models also opens doors to future work which can exploit more complex metadata such as those generated by interactions between users in social media.

Mixed data models take advantage of the flexibility of neural networks to concatenate layers that ingest and process text with traditional numeric features. In simple models, such as the Bag-of-Words (BOW), numerical features are simply concatenated to the text-based data (see Figure 18).

[14] One such embedding is explained in the following section.

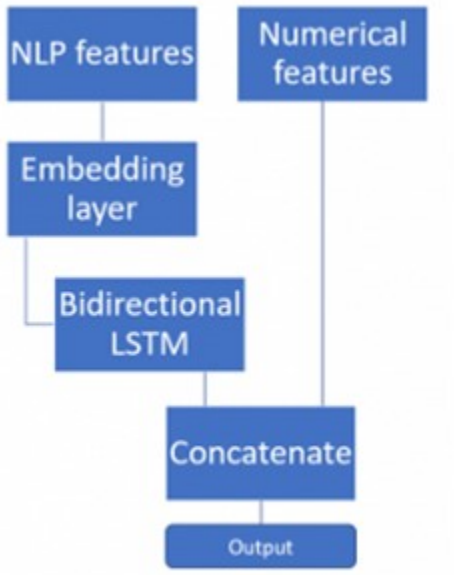
Figure 18 - Mixed-data with simple models as BOW



Source: Freischlag (2020)

Models that employ word embeddings are more complicated to adapt into mixed-data models. This presented an opportunity to extend our model to numeric data and use more than one embedding simultaneously. Our model requires concatenating the text-based feature layer and the numerical feature layer, as depicted in Figure 19.

Figure 19 - Mixed-data for complex models, Embeddings and LSTM



Source: Freischlag (2020)

Exploiting the metadata

In addition to the review text, our enhanced dataset includes numeric, date, and string variables. In total, we created 1,016 additional features detailed below in Table 2.

Table 2 - Feature engineering from metadata

Original feature	Extracted feature
Date the review was done	Day
	Month
	Year
Medical condition: Text	One-hot-encoding of conditions (~900 variables)
Review: Text tokens	Characters count
	Word count
	Word density
	Average word length
	Average sentence length
	Punctuation count
	Upper-case count
Review: Part-of-speech tokens	Noun count
	Verb count
	Adjective count
	Adverb count
	Pronoun count

Compiled by authors.

Combining embedding

In addition to the mixed attribute layers that let us combine different types of features, we decided to explore the model's ability to use more than one word embedding. The first embedding used was the general-purpose GloVe. This embedding comes from a model for distributed word representations constructed by unsupervised learning techniques to obtain the vector representations of the words. We used the publicly available version of pre-trained word vectors "Wikipedia 2014 + Gigaword 5" with vectors of 300 columns.

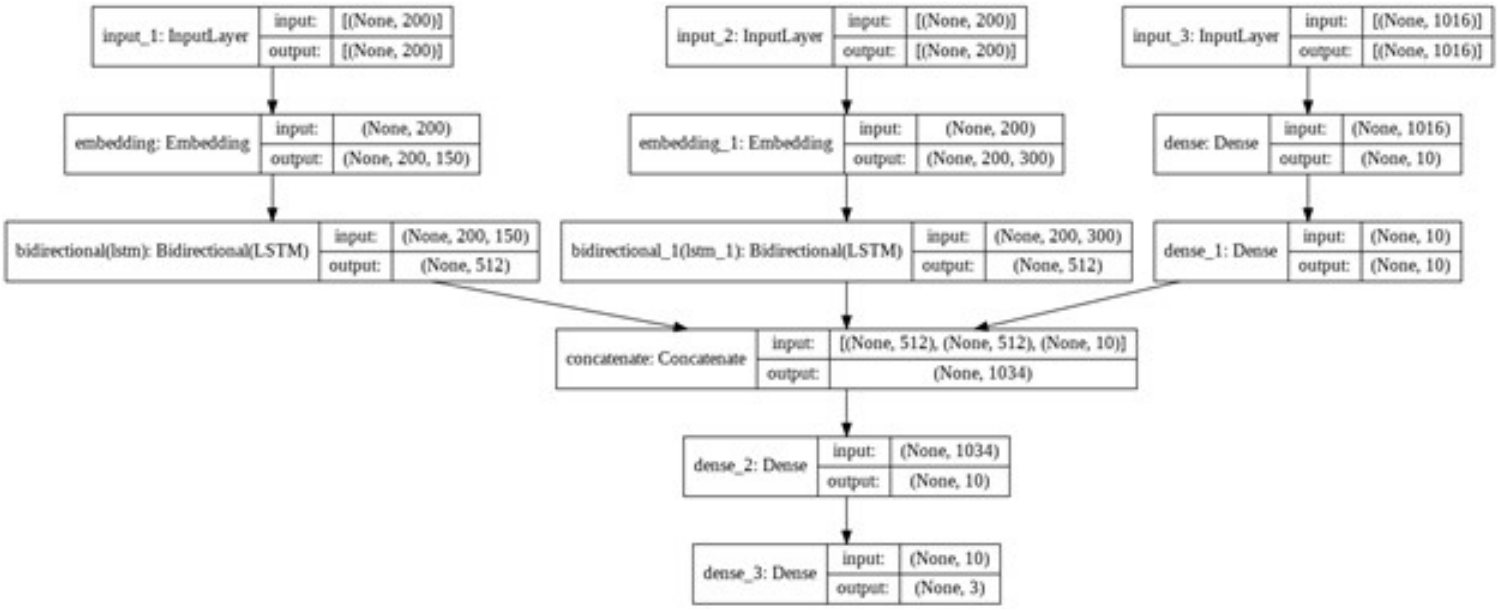
Even though GloVe is a widely used embedding with promising results, we decided to explore domain-specific embedding developments in the drug literature. For example, Colon-Ruiz and Bedmar's (2020) study uses specific embeddings trained for drug reviews. Because traditional NLP models have struggled to make sense of informal language and user-expressed medical concepts, domain-specific embeddings theoretically can provide models new relevant information. This drug-specific embedding was constructed by extracting tweets that mention drug reactions, thus representing colloquial language used to describe drugs and adverse reactions used in our dataset. This complementary embedding that we call "Twitter embedding" was trained using more than a million sentences in English from tweets about drugs. It contains a vocabulary of 26,278 lemmatized words with vectors of 150 columns.

The Model

Considering these steps, we trained our mixed data model with three parallel layers. The first two layers use the text reviews as inputs, transformed by one of the nodes' embeddings. Afterward, the embeddings are fed to two Bidirectional LSTM layers with 256-neuron and different dropout techniques. The third parallel layer takes as an input the 1,016 numeric features and provides these variables to two subsequent dense 10-neuron layers. Subsequently, the three intermediate outputs from these layers are concatenated into a dense 1,034-neuron layer. This number is obtained by summing the 256 neurons of each LSTM twice, as they are bidirectional, plus the 10-neurons from the numeric layer (256*2+256*2+10). Figure 20 displays a visual representation of the model, wherein the left layer corresponds to the text data and Twitter embeddings, the middle layer to the text data with the GloVe embeddings, and the right layer to the numerical feature component.

We trained the final model using a cross entropy loss and an Adam optimizer. The max length of the reviews was set to 200 words, based on the distribution of the review length. The implementation includes both embeddings as trainable, which considering the neurons from subsequent layers adds up to 25,393,585 trainable parameters. The model was constructed and trained using Keras and Google Colab Pro. We included different dropout strategies to address overfitting concerns. However, this dramatically increased the training time. Without any dropouts, the model takes seven minutes per epoch; using regular dropouts, the training time rose to 30 minutes per epoch. When we included recurrent dropout, the computation took 180 minutes per epoch. Given this and the model's behavior, we decide to train it for only ten epochs.

Figure 20 - Mixed data final model



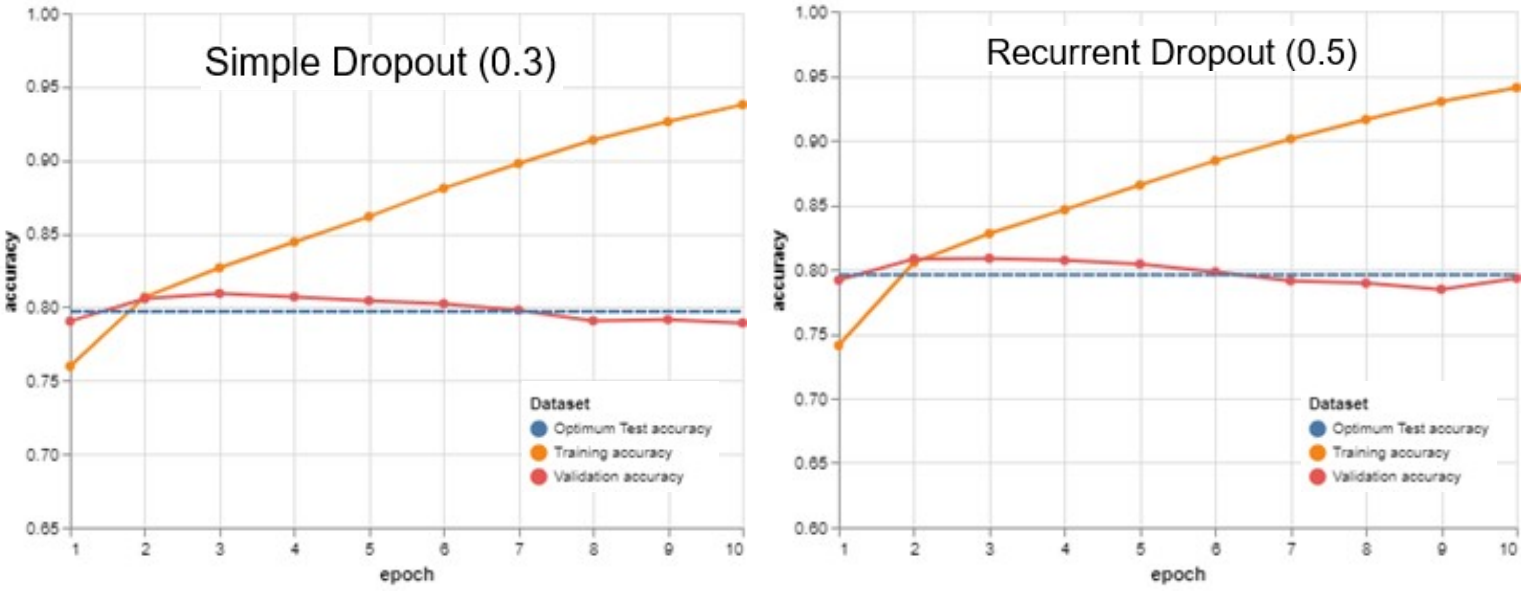
Compiled by authors

Results

Our ambitious implementation and high complexity will tend towards overfitting as the literature suggests(Hastie et al., 2009). To address this concern, we used simple dropouts, recurrent dropouts, and a combination of both in training as suggested by Goldberg (2017).

We present our preliminary results in Figure 21. These models were our first attempts to solve the overfitting detected by the gap between the validation/test metrics and our training accuracy. As we see from the chart, using the dropout techniques in isolation does not improve the situation. The recurrent dropout is the only technique that provides a semblance of improvement.

Figure 21 - Preliminary models results with dropouts

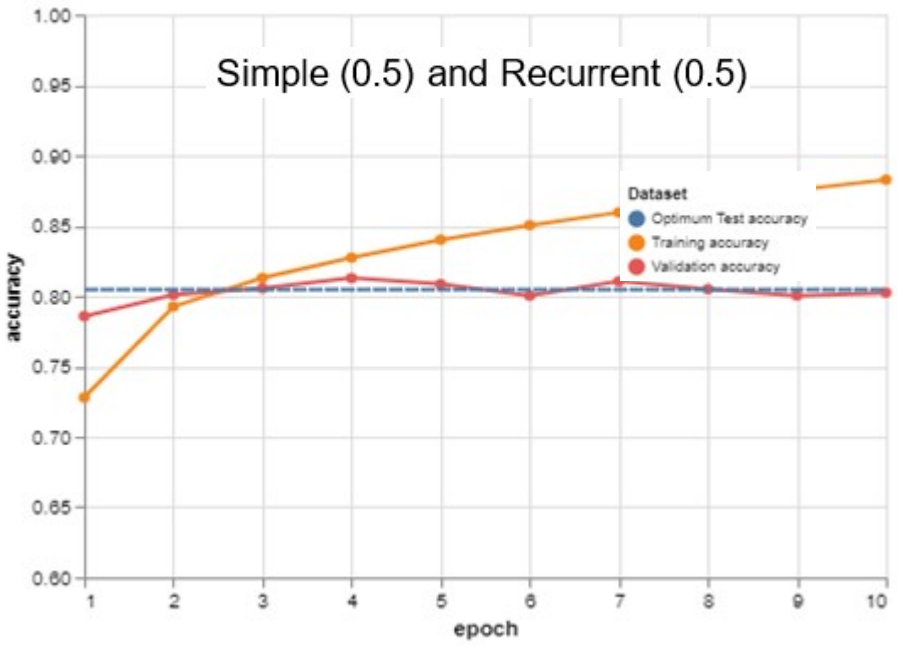


Compiled by authors.

Given this overfitting problem, we combined both the simple and recurrent dropout techniques in the same multi-data model. This variational RNN dropout method of Gal & Ghahramani (2016) is the current best practice for applying dropout in RNNs (Goldberg, 2017).

As we observe from Figure 22, combining both dropout techniques reduces the overfitting level seen in the previous models. The gap from the validation and training accuracies at the 10th epoch for this model is below ten percentage points, while at this stage in the previous figures, the gap is nearly 15 percentage points. This is our chosen best mixed-data model as it obtains a test accuracy of roughly 0.8, a weighted F1 score also of 0.8, and a Cohen's Kappa of 0.63. In addition to attaining the best scores, overfitting is much less pronounced in this implementation

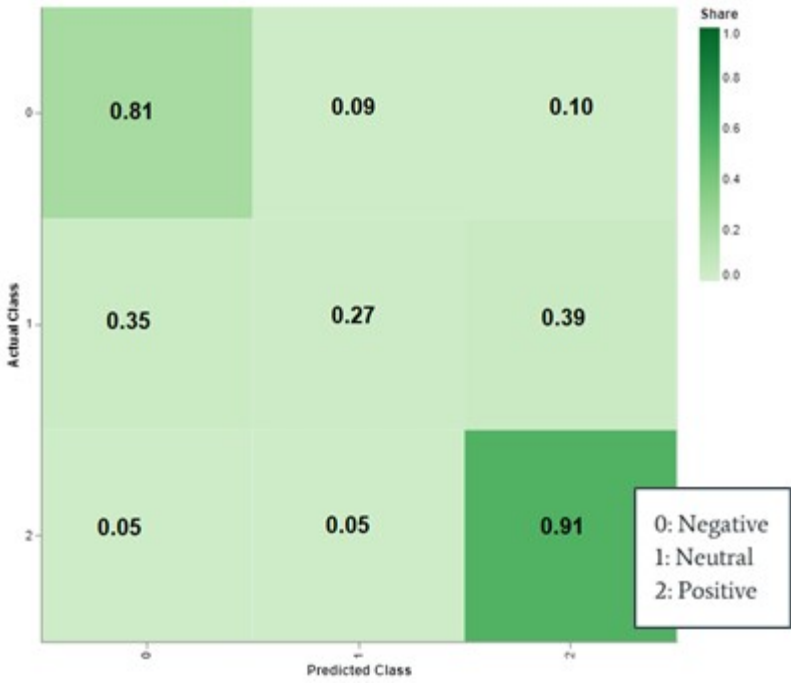
Figure 22 - Mix data final model with both dropouts



Compiled by authors

Finally, we computed the confusion matrix for the predicted and true labels in our test data set, presented in Figure 23. From the image, one can infer that prediction errors are very similar to models from Colon-Ruiz and Bedmar (2020) and the models from the previous sections that only used text as an input. The matrix shows us that positive and negative classes are correctly predicted with accuracies of 0.91 and 0.81. Yet only 27% of the Neutral reviews are correctly labeled by the model. These results should be analyzed in light of the high-class imbalance of the (cleaned and enhanced) dataset. The negative reviews comprise 27% of the dataset whereas neutral reviews only make up 12% of the sample.

Figure 23 - Confusion matrix for selected mixed data model



Compiled by authors.

Limitations and future work

There are several ways in which our models may be improved. Below is a list of some ideas for future research of using drug reviews to predict patient sentiment:

Leakage Analysis

1. Replicate the exact results obtained by Grasser et al. (2018) using their team's code. Recompute the models with the various datasets to get a measure of the leakage effect.
2. Replicate the exact results obtained by Colon-Ruiz and Bedmar (2020) for LSTM + CNN and Biseda and Mo (2020) for the case of BERTs. Obtain a measure of the leakage effect.

Sentiment Prediction

1. Extend our models to include Convolutional layers in our mixed data model. Moreover, the addition of multi-channel convolutions using the text tokens and the part-of-speech tokens in parallel could enhance model accuracy.
2. Include more strategies to deal with the overfitting of the mixed data model. This includes using other dropout techniques, early stopping techniques, and clever feature selection mechanisms.

3. Include an analysis of the imbalance of sentiment categories in the dataset and how this may be affecting our results. Explore techniques as under-sampling, SMOTE,[15] and other weighted metrics.

Other extensions and ideas

1. Include a spelling correction step in the pre-processing task.
2. Test and evaluate if our models successfully predict the sentiment of other reviews, such as online retail reviews. Explore the transfer learning literature.
3. Incorporate name-entity recognition into feature engineering
4. Analyze the potential use of BERTs to exploit predictive power by long-term attention mechanisms.

[15] Synthetic Minority Oversampling Technique

References

Biseda, B., Mo, K. (2020). Enhancing Pharmacovigilance with Drug Reviews and Social Media.

Colón-Ruiz C., Segura-Bedmar I. (2020) Comparing deep learning architectures for sentiment analysis on drug reviews.

Freischlag, C. (2020) Combining numerical and text features in deep neural networks. Towards Data Science. Available at: https://towardsdatascience.com/combining-numerical-and-text-features-in-deep-neural-networks-e91f0237eea4

Gal, Y., & Ghahramani, Z. (2016). A theoretically grounded application of dropout in recurrent neural networks. Advances in neural information processing systems, 29, 1019-1027.

Goldberg, Y. (2017). Neural network methods for natural language processing. Synthesis lectures on human language technologies, 10(1), 1-309.

Grasser, F., Kallumadi, S., Malberg, H., Zaunseder, S. (2018). Aspect-Based Sentiment Analysis of Drug Reviews Applying Cross-Domain and Cross-Data Learning.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media.

Kaufman, S., Rosset, S., Perlich, C. (2011) Leakage in data mining: formulation, detection, and avoidance.

Lie, S., Lee, I. (2019) Extracting features with medical sentiment lexicon and position encoding for drug reviews

Ouyang, X., et al (2015) Sentiment Analysis Using Convolutional Neural Network