Comparing regression and LSTM sentiment analysis models

Language Processing II exam

Jens Egholm Pedersen
xtp778@sc.ku.dk

June 16, 2017

Contents

1 Introduction

Due to the enormous rise in accessible data and processing capabilities, computers have been moving from the restricted formalized domain of mathematics and computability, and into the more complex and seemingly chaotic domain of natural language (Nilsson, 2009; Jurafsky and Martin, 2000). Automating understanding of natural language have a wide range of helpful applications, ranging from translation to recommender systems to generic personal assistants (Cox, 2005; Bird et al., 2009).

As a part of this development, sentiment analysis has been attracting increased attention (Bird et al., 2009; Jurafsky and Martin, 2000).

In the present paper, sentiment analysis is applied to a corpus of movie reviews presented by Pang and Lee (2005). By using standard natural language processing tools and machine learning techniques, four models are built and trained to predict the rating of a movie review on a discrete normalized scale.

2 Background

This section illuminates the theory and background for language processing and basic methodology for processing text. Sentiment analysis as a fundamental concept for this setting will be introduced second, followed by fundamental statistical metrics. Lastly, recurrent neural networks will be introduced before moving on to the data generation.

2.1 Language processing

Following the revolutionizing formalizations of grammar by Chomsky (2002), first written in 1957, much work was put into working with the computability of language (Jurafsky and Martin, 2000). In the beginning of the 21st century, the previous work converged with the statistical machine learning community, giving rise to supervised, semi-supervised and even unsupervised models of language understanding and translation (Jurafsky and Martin, 2000).

There are several approaches to language processing, but it is common to deconstruct text into its constituents and assign meaning to each piece, with the hope that the disassembly gives additional meaning that can aggregates to an understanding of the text as a whole when the pieces are put back together (Jurafsky and Martin, 2000). *Constituents* can range from morphemes to words to whole sentences in this context. This processes is referred to as feature extraction due to the enrichment of the text with meta-data.

A fundamental method for feature extraction is to construct a probabilistic model by simply joining items (ranging from phonemes to word pairs) together in pairs of one, two or more, called n-grams (Jurafsky and Martin, 2000). The advantage of this approach focuses on speed and the simple, yet efficient, probabilities in for instance word pairs which can be applied to predict likelihood of future occurrences (Jurafsky and Martin, 2000).

Term frequency, inverse document frequency (tf-idf) is another relevant feature that describes the relevance of an item, weighted against the number of times it appears in other documents (Jurafsky and Martin, 2000) (see figure 1). Tf-idf favors words that is used heavily in a single document (assumed to be

$$tf - idf = Nw \cdot log(\frac{N}{k})$$

Figure 1: Formula for tf-idf.

important) while diminishing the importance of words that often occur in other documents, and is a heavily used metric (Jurafsky and Martin, 2000).

2.1.1 Sentiment analysis

Another popular approach for disassembling text called part of speech-tagging (POS tagging), assigns grammatical categories to each word and attempt to build a linguistic model that can be understood by a machine (Jurafsky and Martin, 2000). This works well for information retrieval, where objective truths can be extracted[1], but fails to capture the complexity and emotional variety of sentences that appear outside the grammar, such as emotions and opinions[2] (Jurafsky and Martin, 2000).

Sentiment analysis focuses on extracting information about the emotions or opinions of a text (Pang and Lee, 2008). Defining what a *sentiment* is not a simple task, and have been scrutinized extensively (Jurafsky and Martin, 2000; Pang and Lee, 2008). A much used approach was introduced by Ekman (1992) where he devised six basic emotions, which could be conveniently operationalized by computers. Since the dataset for this paper revolves around opinions in a single dimension, *sentiment* will in this paper simply refer to a subjective experience, discretized to a number (see section 3.1).

2.2 Linear regression

Linear regression is a mathematical function that describes how the mean of the output changes with the value of the input (Agresti and Finlay, 2008). The model is appropriate when a linear correlation between the input and output can be assumed, so an increase in the input results in a linear in- or decrease in the output, represented by a straight line in a 2-dimensional coordinate system (Agresti and Finlay, 2008).

2.2.1 Error reporting

The predictions made by linear regression model (y), does not always fit the actual values ($\hat{y}$). To describe this error and to give indicators on the quality of the model, the mean absolute error (MAE) and root mean squared error (RMSE) is introduced (Agresti and Finlay, 2008; Chai and Draxler, 2014).

MAE in figure 2 describes the mean of the absolute error values. If a model has an MAE of 1, the average distance to the actual value from the predicted value is 1. The bigger the mean error value, the worse the model is.

[1] Such as the objective statements about an object in the sentence "*The cat likes tuna fish*".

[2] One example is a heavily sarcastic sentence like "*I'd really truly love going out in this weather!*". For POS-tagging sarcasm is difficult to capture because it is not implicitly encoded in the grammatical categories.

$$MAE = \frac{\Sigma|e|}{n}, e = y - \hat{y}$$

Figure 2: Mean absolute error (MAE).

$$RMSE = \sqrt{\frac{\Sigma(e)^2}{n-2}}, e = y - \hat{y}$$

Figure 3: Root mean square error (RMSE).

However, this figure does not capture the error distribution of the model (Chai and Draxler, 2014). If a Gaussian distribution is assumed for the errors, it is important to know the standard distribution to indicate *how* consistently wrong the model is (Agresti and Finlay, 2008). The root mean square error measures the standard deviation in the sample of the errors, making it a good description for the model accuracy, compared to other models (Agresti and Finlay, 2008; Chai and Draxler, 2014).

Each metric should not stand alone however, and MAE and RMSE is typically reported together (Chai and Draxler, 2014).

2.3 Neural networks

Inspired by the biological brain, models for neural networks have been applied in computer science the 1950s, and have become integral part of machine learning (Nilsson, 2009; Russell and Norvig, 2009). Neurons are essentially functions that operate on some input and respond with some output (Russell and Norvig, 2009). The inputs for a neuron are weighted to give different input channels - or input dimensions - varying significance. These weighted inputs arrive to an activation function that decides whether the neuron should 'fire' or not (Nilsson, 2009). Sigmoid or hyperbolic tangent are popular activation functions for numerical predictions because of their steep logistic properties while retaining differentiability. Neural networks excell by their adaptability and have long been applied to the field of language processing (Jurafsky and Martin, 2000).

Simple neural networks can be stacked in layers, where each neuron will assign weights to the input, to determine its significance for the activation function (Nilsson, 2009). By tuning the weights of the neurons, such groupings can learn to fire on certain input patterns (Russell and Norvig, 2009). However, this structure have proved brittle and difficult to train because the neurons do not retain their weights for long (Nilsson, 2009; Russell and Norvig, 2009). Rumelhart et al. (1988) suggested a method to avoid this instability by adjusting the weights of the neurons in retrospect with a method called back-propagation (Rumelhart et al., 1988; Nilsson, 2009). This optimization is operationalized by a function that can describe the *loss* of effiency in a network, and then adjust the weights correspondingly using gradient descent (Russell and Norvig, 2009).

2.3.1 Long short-term memory networks

Back-propagating networks with a high complexity in layers and neurons have shown to be hard to optimize above a certain point, because the gradient de-

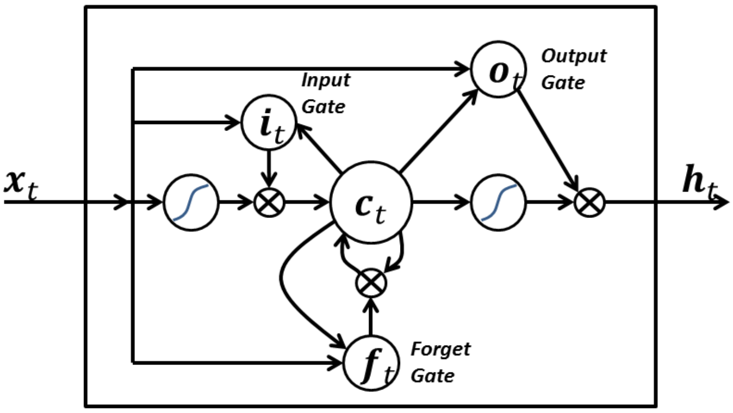

Figure 4: An *peephole* LSTM unit with three gates (input, output and forget).

scent algorithm are inefficient at escaping local extrema in a high-dimensional space (Russell and Norvig, 2009). Recurrence have been used to counter this problem, by conserving or returning to previous, optimal values (Schmidhuber, 2015; Russell and Norvig, 2009). Hochreiter and Schmidhuber (1997) suggested a certain type of memory-retaining networks that is able to balance between both storing previous optimal state, while also forgetting context when new and different input arrives (Hochreiter and Schmidhuber, 1997; Schmidhuber, 2015). They dubbed the model long short-term memory (LSTM) networks, to capture both properties (Hochreiter and Schmidhuber, 1997).

The LSTM model seen in figure 4 builds on the idea of a more complex neuronal unit, with several *gates*, collaborating to give the LSTM the desired property (Gers, 2001). A traditional unit contains three gates: an ”input” gate that control the degree with which new information should influence the current memory state, a ”forget” gate that decides whether a value should be retained and an output gate that decides how much influence the memory should have on the activation of the unit (Hochreiter and Schmidhuber, 1997; Gers, 2001). There are a plenitude of other architectures, but for this paper, the LSTM model shown in figure 4 is used (Gers, 2001).

2.3.2 Optimization and regularization

Recurrent neural networks are typically trained with a continuously applied loss function and gradient descent for weight correction (Russell and Norvig, 2009; Hochreiter and Schmidhuber, 1997; Schmidhuber, 2015). Stochastic gradient descent and resilient back-propagation are two common approaches, which have proved efficient (Russell and Norvig, 2009; Nilsson, 2009). This paper applies a third method, based on the RMSE algorithm in figure 3, called root mean square propagation (RMSprop), which preserves the stochastic properties while controlling for previous errors in the network (Tieleman and Hinton, 2012).

LSTMs and machine learning models in general can overfit to the data they are trained on, loosing their ability to predict data outside the training set (Russell and Norvig, 2009; Nilsson, 2009). A common approach to avoid overfitting is to introduce additional information in the learning process, so the model gets sufficiently confused to lose any overfitted optimizations, but sufficiently focused to only save the relevant parameters (Schmidhuber, 2015; Nilsson, 2009). For LSTM networks a common approach is simply to set a random part of the data to zero (Schmidhuber, 2015). This type of data dropout is applied later in the paper.

2.3.3 Error reporting

Similar to the linear regression model, the LSTMs are predicting a numerical value (see 3.1). For this type of prognoses MAE and RMSE are commonly used as indicators of the model accuracy (Russell and Norvig, 2009; Nilsson, 2009; Schmidhuber, 2015). As with the linear model, they are capable of capturing both the average error in absolute terms, and the deviation from the mean of the expected predictions.

3 Data generation

Following the theoretical background of the paper, this section zooms in on the prediction models and data set, while finally reporting on the results of testing the models againts the data.

3.1 Data set

The data is a movie review corpus, where four reviewers rate movies in 5010 reviews (Pang and Lee, 2005). The ratings of each reviewer is aggregated to create three rating data sets: one with three categories, one with four categories and a discrete, normalized scale from 0 to 1. The aggregation were made with a technique called metric labelling, which attempts to take label preference of reviewers into account, essentially controlling for more or less positive reviewers compared to the mean (Pang and Lee, 2005). This method increases the external validity of each reviewer and gives a better basis for comparison.

Of the three different scales this paper concentrates on the discrete ratings. The main reason for this is firstly the increased granularity of the scale, giving a more correct view of the aggregation of the different scale. Second, sticking to one scale rather than several makes it easier to compare results across models. A good accuracy in one model does not necessarily imply superiority over others, if the first model have fewer categories. Using a discrete scale makes it easier to report numerical measures and provide a more accurate picture of the pros and cons of each model.

3.2 Preprocessing

During the preprocessing step the data was split into a training and a test set using k-fold cross validation (Russell and Norvig, 2009). To ensure that the model does not overfit to a single review author, the k-fold was done once per reviewer, each time excluding one reviewer. This resulted in a 4-fold split where each split completely excludes one of the four reviewers. Each test performed on the data is thus done on a single reviewer the model has not seen before in the training step.

Model	MAE	RMSE
Linear	0.1649	0.2075
Linear + tf-idf	0.1494	0.1855
LSTM	0.1667	0.2045
LSTM + tf-idf	0.1641	0.2014

Figure 5: Comparison of the MAE and RMSE for the four different models.

Lastly the data was split into text as the input values (x) and ratings as the output values (y) and compressed to a file for the next processing step (see appendix B).

3.3 Prediction models

To process the data two types of models were built: one with Scikit Learn Sklearn) and one with the deep learning library Keras (see appendix B). These two types represent a simple linear baseline, and a more advanced adaptive machine learning approach.

For each of the model types the impact of using tf-idf is examined by creating two implementations: one using the tf-idf algorithm and one without. This results in four models in total that can be compared within the same type (using tf-idf or not) and between types.

All four models are based on bi- tri- four- and five-grams (2-5), created before the tf-idf process Jurafsky and Martin (2000). Unfortunately the maximum number of features for the n-grams was restricted to 10'000 due to memory limitations.

For the LSTM models, a pipeline was constructed with 64 LSTM units, a dropout rate of 50% and finally a single neuron neural network with a sigmoid activation function to provide the single-dimensional rating value. The model was optimized using RMSprop and each model was given 8 run-throughs to train the network.

The source code for the models are available in appendix B.

3.4 Model evaluation

The results are presented in figure 5. Each model was run four times (once for each k-fold split), and the reported metrics are the result of averaging the MAE and RMSE metrics over the four iterations.

4 Quantitative evaluation

The results for the LSTM model are slightly worse than the linear. With a MAE of .1667, the LSTM model improves to .1641 with tf-idf scores. That is, on average the model misses with either .1641 too high or too low on the rating scale. The average error margin is then .3282. Taking into consideration that the scale ranges from 0 to 1, .3282 is a high number. In comparison the MAE of the linear model begins with .1649 and improves to .1494.

The RMSE is also less in the linear model, dropping to .1855 with tf-idf scores. This points to a lower error standard deviation, meaning that the predictions falls closer to the mean than the other three models.

For all four models the MAE scores are within a small margin between .1494 and .1667. Similarly the RMSE values are within a minuscule interval of .078.

With a small difference in the MAE of 0.015 between the LSTM tf-idf model and the linear model with added tf-idf scores, the results show that the linear tf-idf model has the best predictive abilities.

5 Qualitative evaluation

The very definition of *sentiment* in section **??** as a subjective opinion makes comparison difficult. The effort presented by Pang and Lee (2005) to normalize this across reviewers is commendable, but it is difficult to measure their success. The paper introduces the idea of a positive sentence percentage (PSP), which is used to evaluate how positive a sentence is based on the positive words in it. Such a method requires a positivity value of each word, which again is an opinionated source of data.

Machine learning algorithms are known to require large amounts of data before it produces useful predictions (Schmidhuber, 2015; Russell and Norvig, 2009). In the present data set there are only four reviewers. They produce over 5000 reviews, but the scores presented in figure 5 are based on a k-fold that excludes each reviewer. In particular the LSTM model would most likely perform better with a higher number of reviewers to train on.

The n-grams features ran into an unfortunate memory limitation, where only 10'000 features could be evaluated at a time. Languages have a much higher degree of complexity, and removing this ceiling would certainly make the results more accurate.

Criticism aside the results are consistently within a small margin, indicating that it might be hard to achieve much better results with the available data. Despite the difficulty of comparing opinionated data, this is an encouraging result. All else being equal, the best model is capable of approximating the opinion of a piece of text on a linear scale. This would surely be easier for categorical values, but the discrete scale contains more information and is easier to generalize to other contexts.

6 Conclusion

Using a dataset of 5010 movie reviews from four reviewers this paper applied four sentiment analysis models, based on linear regression and LSTM machine learning, to predict the ratings of the movie reviews. While the LSTM models were close to the performance of the linear models, the linear models proved superior.

It was concluded that the LSTM model might suffer from the small amount of reviewers, and that future work will have to be done on a larger machine with more memory.

Despite the weaknesses the two completely different models provided results that fell within a small margin, hinting at a dataset that might be hard to extract much more information from.

There are a number of directions future work could go. It would first and foremost be interesting to examine other techniques like enrichment through a database of world-knowledge. In for instance the AFINN database, individual words are given a score to describe their positive or negative meaning (Nielsen, 2011). This addition allows the models to improve the input word by word, possibly improving the predictability of a review. However, as discussed in section 2.1.1 these models does not always capture complex phenomenons like irony and are not freed from individual opinions.

Another feature extraction technique that have seen a surge in popularity is word2vec, where words are extracted into a multidimensional vector space (Schmidhuber, 2015). This opens up for interesting geometrical analyses because words can be seen as points in space. For instance the meaning of one word in relation to another can be expressed as the euclidean distance (**?**Schmidhuber, 2015).

Second, it could be interesting to look into other LSTM type networks with a different unit design (Schmidhuber, 2015) or perhaps even deep learning techniques like meta-cognitive representations capable of self-introspection, which would provide a much stronger loss function heuristic (**?**).

7 References

Agresti, A. and Finlay, B. (2008). *Statistical Methods for the Social Sciences (4th Edition)*. Prentice Hall, 4 edition.

Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly, Beijing.

Chai, T. and Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)? - Arguments against avoiding RMSE in the literature. *Geoscientific Model Development*, 7:1247–1250.

Chomsky, N. (2002). *Syntactic Structures*. Mouton classic. Bod Third Party Titles.

Cox, M. T. (2005). Metacognition in computation: A selected research review. *Artificial Intelligence*, 169(2):104 – 141. Special Review Issue.

Ekman, P. (1992). An argument for basic emotions. *Cognition and Emotion*, pages 169–200.

Gers, F. (2001). Long short-term memory in recurrent neural networks.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Jurafsky, D. and Martin, J. H. (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition.

Nielsen, F. Å. (2011). Afinn.

Nilsson, N. J. (2009). *The quest for artificial intelligence - A history of ideas and achievements*. Cambridge University Press.

Pang, B. and Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of ACL*, pages 115–124.

Pang, B. and Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2:1–135.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA.

Russell, S. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude.

8 Appendix A: Data and preprocessing

The data used in this report is collected from Pang and Lee (2005). The data is available in the software repository referenced in appendix B.

A readme file with an in-depth description on the data source is available at:

http://www.cs.cornell.edu/people/pabo/movie-review-data/scaledata.README.1.0.txt

9 Appendix B: Software

The data preprocessing, model building and model evaluation was done using Python 3. All software presented in this report is open-source and available through GitHub, along with an in-depth readme file on how to reproduce the results:

https://github.com/Jegp/langprocexam

Below follows a list of external software packages and frameworks used in the project.

9.1 NLTK

A toolkit for natural language processing in python. Visited 15th of June 2017.

http://www.nltk.org/api/nltk.sentiment.html

9.2 Scikit-learn

Machine learning library for Python. Visited 15th of June 2017.

http://scikit-learn.org/stable/index.html

9.3 Tensorflow

A machine learning library initially developed by Google Inc. Visited 15th of June 2017.

https://www.tensorflow.org/

9.4 Keras

A deep learning library which builds on top of Tensorflow. Visited 4th of June 2017.

https://keras.io

10 Appendix C: Figure sources

10.1 Figure 4

Source: Wikipedia.org
https://en.wikipedia.org/wiki/File:Peephole_Long_Short-Term_Memory.svg.