Technical University of Denmark

42186
Model Based Machine Learning

Sarcasm Detection

Authors:

Nikolas Thuesen – $s152993$
Theis Hjortkjær – $s163700$
Christopher Jensen – $s164272$

May 29, 2020

Department of Management Engineering

1 Introduction

Sarcasm and irony are commonly used in written and spoken language but can be difficult to detect - sometimes even for humans. This project attempts to distinguish newspaper headlines from ”The Onion” (satirical) and HuffPost (non-satirical) using a probabilistic model-based machine learning approach. The data was obtained from a Kaggle Challenge [1]. More specifically, we implement a hidden markov model used for classification. This model did however not perform satisfactory, and we therefore also implemented and tested number of additional models. All programming was done in Google Colab, and modeling was done using Pyro [2].

1.1 Preprocessing

Text data can be challenging for a model to understand, and we therefore used word embedding (GloVe, Global Vector Representation [3]) to represent the data. Each word is transformed into a 100-dimensional vector, which represents the semantics of the word. Before the embedding, we also removed punctuation and stopwords, made all words lowercase and then removed headlines consisting of less than three words.

2 Hidden Markov Model

A hidden markov model (HMM) is used for sequential data, with the goal of finding a ”hidden representation”, also called a latent state, using an observed sequence for each datapoint. In our case each headline has the same latent state throughout the whole headline (sarcastic/non-sarcastic) and the observed values will be the individual words in the headline. When choosing the HMM, we knew that it was not well suited for binary-classification. However, due to the structure of the HMM, it was very natural to implement in Pyro and we saw it as a great way of learning the ways of Pyro.

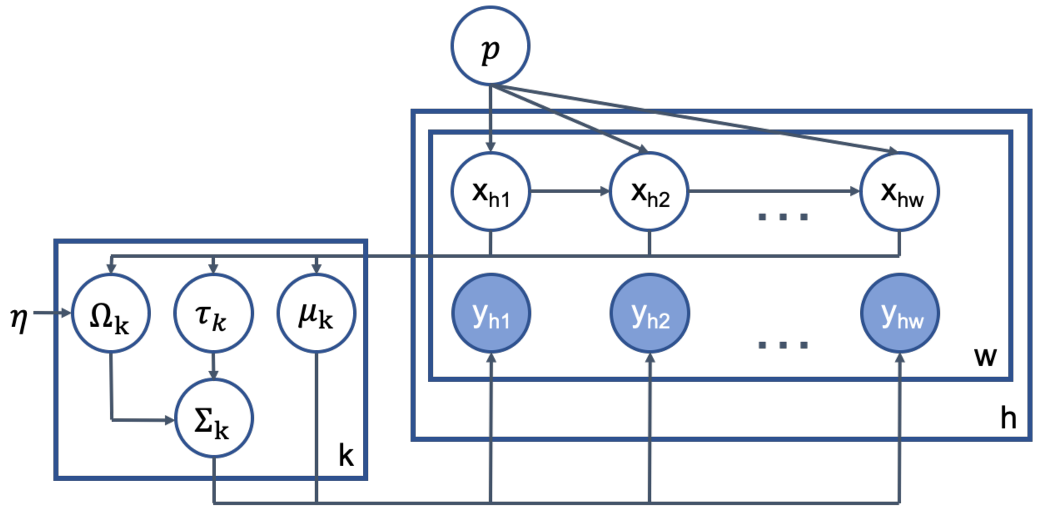

Figure 1: Probabilistic Graphical Model for our HMM.

1. Draw transition matrix coefficients $p \sim Dir(\alpha)$
2. For each state K
   a) Draw mean from multivariate normal $\boldsymbol{\mu_k} \sim \mathcal{N}(\boldsymbol{\mu_k}|0, \boldsymbol{I})$
   b) Draw correlation matrix $\boldsymbol{\Omega_k} \sim LKJCholesky(\eta)$
   c) Draw coefficient scales $\boldsymbol{\tau_k} \sim HalfCauchy(0{,}1)$
   d) Calculate lower triagular representation of covariance matrix:
   $$\boldsymbol{\Sigma_k} = diag\_matrix(\boldsymbol{\tau_k})\boldsymbol{\Omega_k}$$
3. For each headline (H)
   a) Draw state $x_{h(w=1)}$ for first word $x_{h(w=1)} \sim Bernoulli\left(\frac{1}{2}\right)$
   b) For each subsequent word (W)
      i. Draw x_{hw} from transition matrix $x_{hw} \sim p(x_{hw}|x_{h(w-1)})$
      ii. Draw word classification $\boldsymbol{y_{hw}} \sim \mathcal{N}(\boldsymbol{y_{hw}}|\boldsymbol{\mu}_{x_{hw}}, \boldsymbol{\Sigma}_{x_{hw}})$
   c) Assign label to whole headline (mode of all x_{hw})

Figure 2: The Generative Process for our HMM.

Prediction of a new headline was performed by taking the mode of all the latent states in the headline, and assigning it non-sarcastic in case of a tie. This method introduces a bias to the classifier, skewing its predictions towards non-sarcastic labels. Another flaw in our model is that the first words in each headline has a random assignment, but still contributes just as much to the classification as the words towards the end of the headline, where the model has learned more about the sentence. Furthermore, due to the structure of a HMM, the random assignment of the first word significantly affects the prediction of the latent state of the second word and so forth. We attempted to fix this by putting a logistic regression classifier on the latent variable, but due to the headlines not all consisting of the same number of words, we found ourselves unsuccessful.

2.1 Training Using SVI

Our HMM model is trained using Stochastic Variational Inference (SVI). This method tries to approximate the true posterior distribution $p(z|x)$ by finding a distribution $q(x)$ which lies close to $p(z|x)$. It does this by calculating the KL-divergence from $q(x)$ to $p(z|x)$ and minimizing this distance by maximizing the ELBO. As the ELBO increases we can see that $q(x)$ approaches $p(z|x)$.

$$KL(q(\mathbf{z})\|p(\mathbf{z}|\mathbf{x})) = -(\underbrace{E_q[\log p(\mathbf{z},\mathbf{x})] - E_q[\log q(\mathbf{z})]}_{E\mathbf{LBO}}) + \underbrace{\log p(\mathbf{x})}_{\text{log evidence}} \tag{1}$$

2.2 Results and Discussion of HMM

Partly because of the large amount of data, the model took a considerable amount of time to train. It also did not perform well, even when we attempted to change and optimize hyperparameters. When evaluating the performance of our model on a test set consisting of 30% of the orignial data, the predictions were as good as random. This can be seen in the confusion matrix in **Figure 3**. A graph showing how the ELBO is optimized can be found in the appendix.

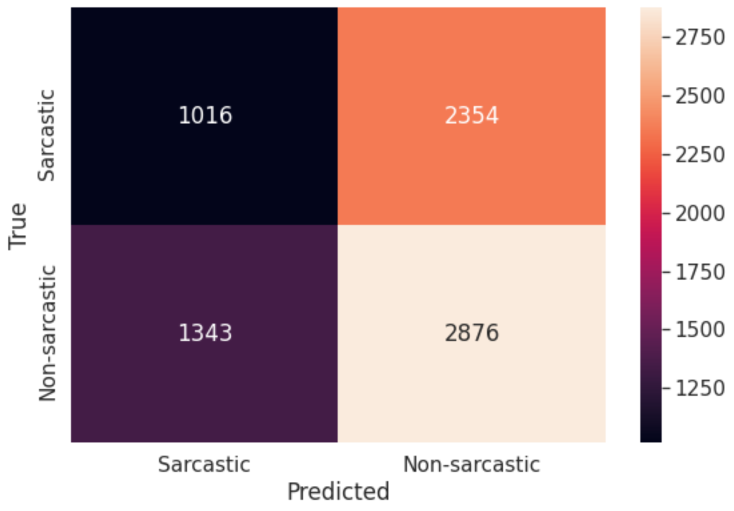

Figure 3: The performance of the HMM model on our test set (30% of observations). The model is biased towards non-sarcastic predictions, but the predictions are as good as random (51.3 % accuracy in the run illustrated here).

3 Additional Models

As a supplement to the poorly performing HMM, we implemented a number of other models. These were (i) a probabilistic recurrent neural network (RNN) tested both with generated data and our real data set. (ii) a standard LSTM network and finally, (iii) a logistic regression classifier. We successfully implemented the probabilistic RNN and it was able to learn on trivial simulated data, however this also fell short in trying to predict sarcasm from our original data set. Our LSTM network implemented in Pytorch[4] was able to learn and had an accuracy of 67.9 % on our sarcasm data set, however this was only evaluated on training data, so it might be a result of overfitting.

The logistic regression classifier was built with the intend to try and simplify the problem at hand. For this, each headline was reduced to 4 critical values given by different similarity measures. Using these similarity measures our, logistic regression with a regularization parameter was trained. The model did not perform spectacularly, but had a prediction accuracy of about 55% (see **Figure 5**). This model is described more thoroughly in the **Appendix B**.

4 Conclusion and Future Work

Throughout this project, we discovered that inferring sarcasm from a sentence of words is a difficult problem - especially for sarcasm detection. The noise to signal ratio is high even though the words are embedded in a way, that is supposed to reveal the meaning of each individual word. However, for sarcasm there is a heavy dependency in the order of words and how words is used in a larger context. This fact makes it extremely hard for both a HMM and RNN to correctly classify sarcasm.

By further feature engineering the embedded words using similarity measures, we showed that a logistic regression can outperform a random guess, and thereby perform better than both our probabilistic HMM and RNN.

As mentioned earlier, one of the major flaws of our approach was to not consider context, which is a core aspect of sarcasm. Our models looked at individual words and used this as input data for sarcasm prediction. For future work, a more suitable model would be able to asses headlines as a whole and evaluating individual words in relation to the whole sentence.

5 References

[1] Rishabh Misra. News headlines for sarcasm detection. `https://www.kaggle.com/rmisra/news-headlines-dataset-for-sarcasm-detection`, 2019. Last accessed 28. May 2020.

[2] Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul A. Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep universal probabilistic programming. *J. Mach. Learn. Res.*, 20:28:1–28:6, 2019.

[3] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[4] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[5] Kevin Patel Pushpak Bhattacharyya Mark Carman Aditya Joshi, Vaibhav Tripathi. Are word embedding-based features useful for sarcasm detection? 2016.

A Training of the HMM

In pyro, instead of maximizing ELBO, the negative ELBO is minimized and the following plot was generated as the model learned:

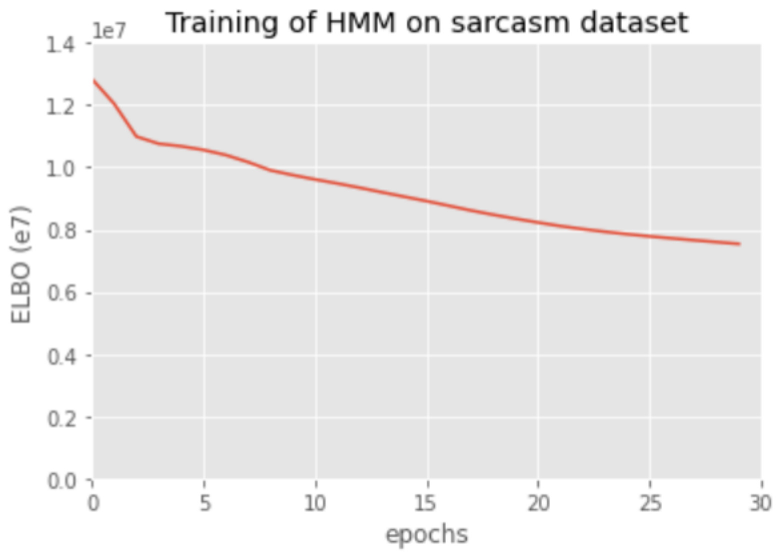

Figure 4: Negative ELBO faling, which results in decreasing of KL-divergence.

B Non-Probabilistic Logistic Regression

This baseline model was created to explore the ability of classifying headlines without the use of a probabilistic model. This model is a simple logistic regression with a regularization parameter, which is based of the idea presented in an article [5].

Essentially what this model does, is compare the words within each headline to each other and extracts critical values. It does this by computing the cosine similarity between all possible word pairs within a headline, and then extracting 4 values: (i) the maximum similarity score for the most similar word in the sentence, (ii) the minimum similarity score for the most similar word in the sentence, (iii) the maximum similarity score for the least similar word in the sentence, and (iv) the minimum similarity score for the least similar word in the sentence. Logistic regression is then trained on these values to classify each headline to be either sarcastic or not.

B.1 Results and Discussion of Logistic Regression Model

The model did not show any spectacular results, with a classification percentage at about 55%, as seen in **Figure 5**. This could partly be due to the fact that whole headlines with words represented as 100 dimensional vectors was reduced to 4 critical values, as well as sarcasm being extraordinarily hard to detect due to the contents of its nature.

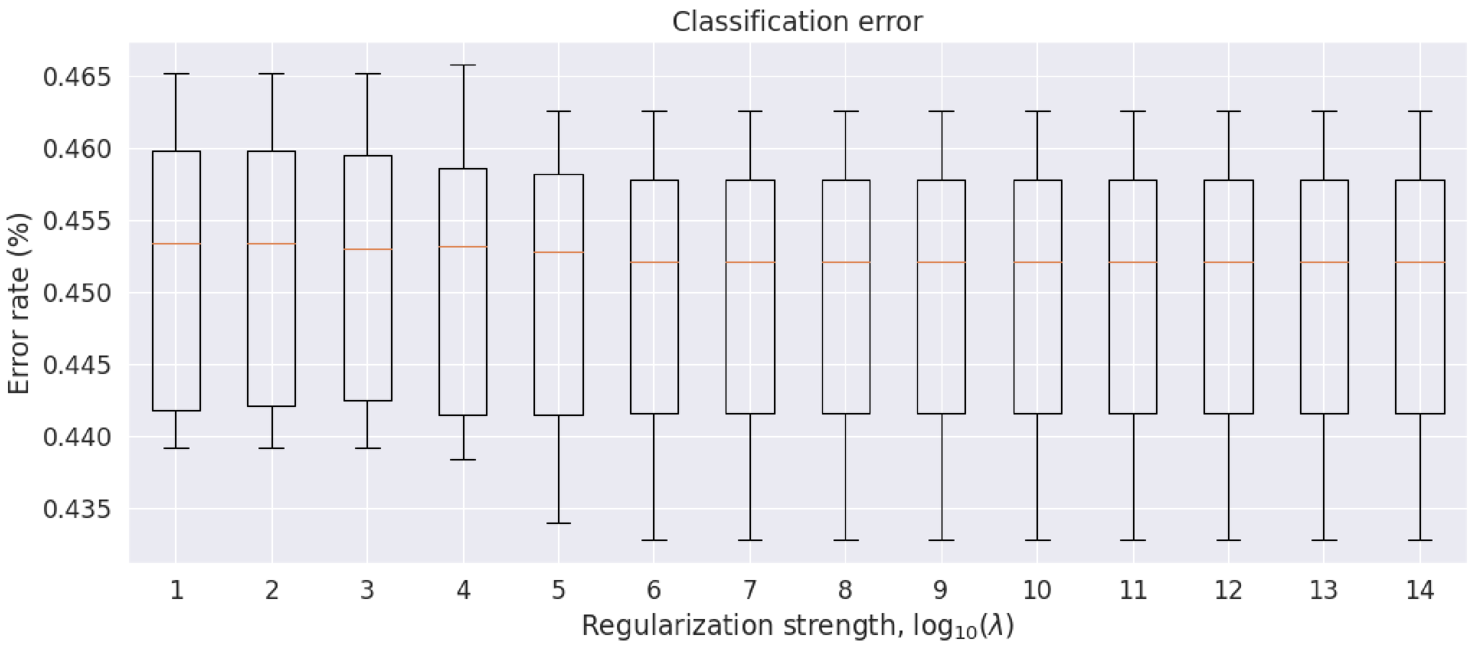

Figure 5: Boxplots of classification errors containing 10 values for each regularization parameter. The red line in the center marks the median for 10 errors in each boxplot.