ARTIFICIAL INTELLIGENCE - FEBRUARY 2021

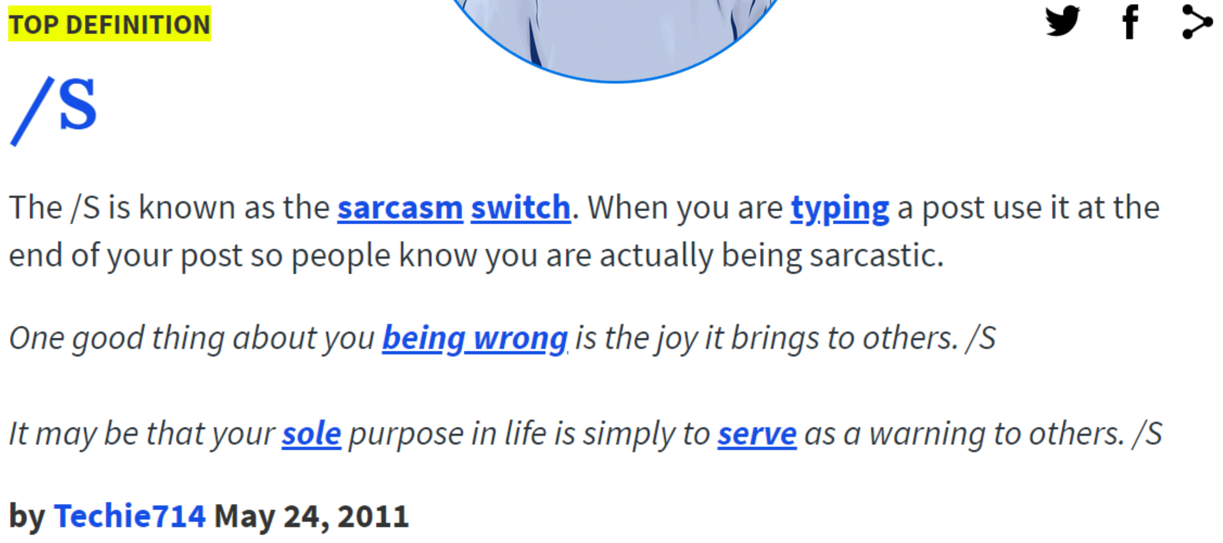
WORD EMBEDDINGS COMPARISON FOR SARCASM DETECTION

FRANCESCO STRANIERI - 816551 - f.stranieri1@campus.unimib.it

FRANCESCO PORTO - 816042 - f.porto2@campus.unimib.it

INTRODUCTION

- **Sarcasm** is the expression of criticism and mockery in a particular context, by employing words that mean the opposite of what is intended
  - **Context** plays a key role in Sarcasm Detection (*context incongruity* [1])
- Different proposed approaches for Sarcasm Detection [2]:
  - **Rule-based**
  - **Statistical-based**
  - **Machine Learning-based**


TOP DEFINITION

/S

The /S is known as the **sarcasm switch**. When you are **typing** a post use it at the end of your post so people know you are actually being sarcastic.

One good thing about you ***being wrong*** *is the joy it brings to others. /S*

It may be that your ***sole*** *purpose in life is simply to* ***serve*** *as a warning to others. /S*

by Techie714 May 24, 2011

RELATED WORKS

The problem of identifying sarcasm has been tackled multiple times in the past:

- In Gonzalez-Ibanez et al. (2011) [3], features like **unigrams** have been used for identifying sarcasm in tweets
- In Maynard and Greenwood (2014) [4] and Liebrecht et al. (2013) [5] features specific to the application itself are used, such as **hashtags** and **emoticons**
- In Pozzi et al. (2016) [6] rule-based approach that use **POS (Parts Of Speech)** have been used
- Felbo et al. (2017) [7] came-up with a new model that uses a **CNN (Convolutional Neural Network)** for the extraction of the sentiment from a sentence using neural networks and later used it to detect sarcasm
- In Misra et al. (2018) [8] a **hybrid** approach has been proposed; it uses an attention mechanism that highlights the words that contribute to sarcasms. Hybrid means that it uses <u>both</u> a CNN and a **Long Short Term Memory (LSTM)** network

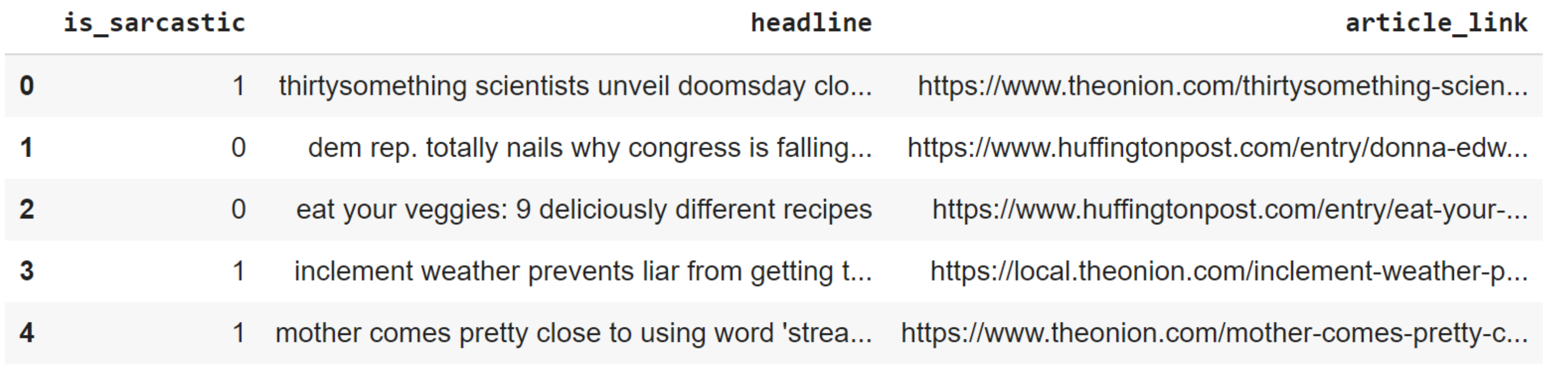
OBJECTIVES OF OUR PROJECT

- In this project, we propose a Deep Learning architecture for **Sarcasm Detection**, while utilizing pre-trained Word Embeddings from three well-known models: *Word2Vec*, *fastText* and *Glove*
- By taking inspiration from CADE, our intuition is to use a non-trainable Embedding layer representing the three models and a Bidirectional Long Term Short Memory (Bi-LTSM) layer
- This model will also be compared to current state-of-the-art approaches for Sarcasm Detection in order to check if we can reach comparable results
- We also try to understand why it works from a semantic point of view. Does the accuracy of sarcasm detection depend on the nature of word embeddings?

DATASET

- We use the **News Headlines Dataset for Sarcasm Detection** (kaggle.com/rmisra/news-headlines-dataset-for-sarcasm-detection)
- Contains 28619 headlines from:
  - **HuffPost**, an actual non-sarcastic journal (huffpost.com/)
  - **TheOnion**, a sarcastic online journal (theonion.com/)
- Advantages of this dataset over other social media datasets:
  - Less **noise** and **inaccuracies,** as headlines are written by professionals
  - Headlines are **self-contained**, no extra context needed

This makes the use of pre-trained Word Embeddings efficient as most of the words will most likely be contained in the embedding



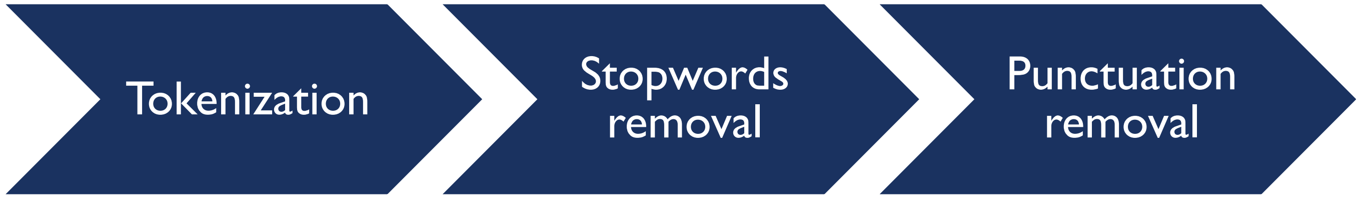
	is_sarcastic	headline	article_link
0	1	thirtysomething scientists unveil doomsday clo...	https://www.theonion.com/thirtysomething-scien...
1	0	dem rep. totally nails why congress is falling...	https://www.huffingtonpost.com/entry/donna-edw...
2	0	eat your veggies: 9 deliciously different recipes	https://www.huffingtonpost.com/entry/eat-your-...
3	1	inclement weather prevents liar from getting t...	https://local.theonion.com/inclement-weather-p...
4	1	mother comes pretty close to using word 'strea...	https://www.theonion.com/mother-comes-pretty-c...

DATASET PREPROCESSING

We perform the following tasks [9]:

- **Tokenization**: the headlines are broken into *tokens*, by using the whitespace as separator
- **Stopwords removal**: *stopwords* are those words that do not provide any useful information, such as *the, is*, *at, which,* and *on*
- **Punctuation removal**: we also remove any *punctuation* symbol

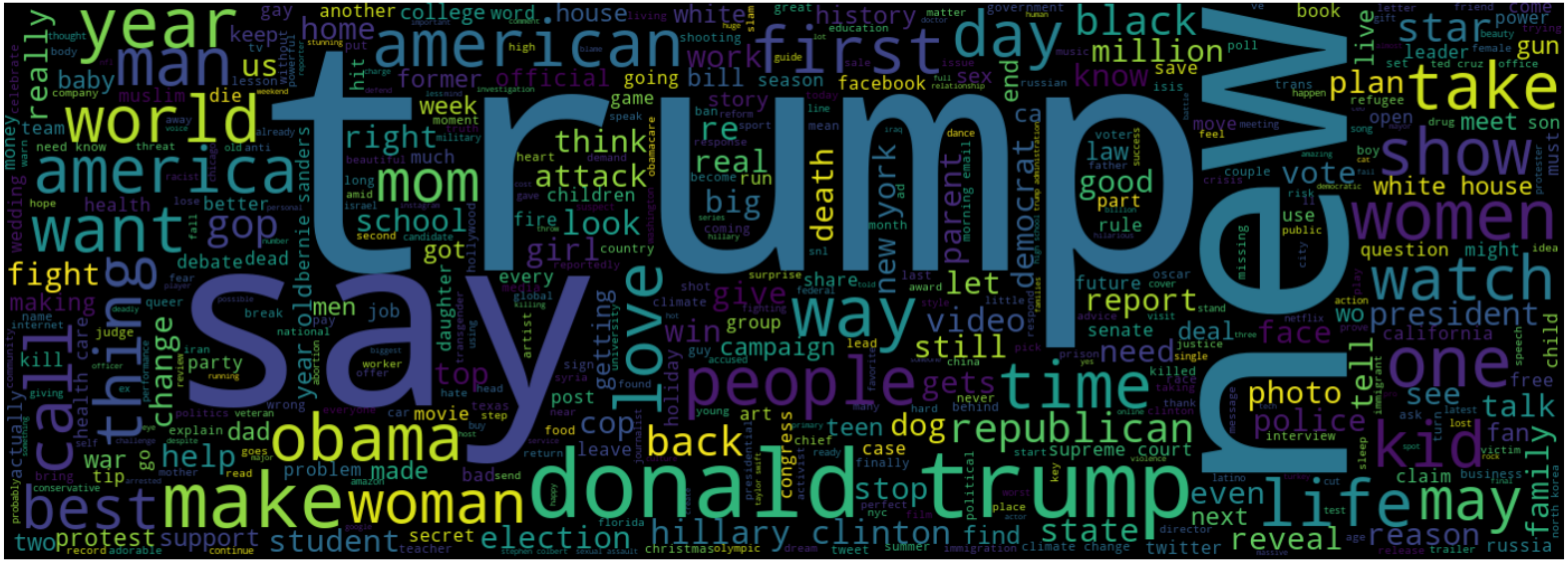
We did not perform **stemming** as we wanted to avoid ***overstemming***



For these preprocessing tasks, we used the **NLTK** library (nltk.org/)

DATASET VISUALIZATION (1)

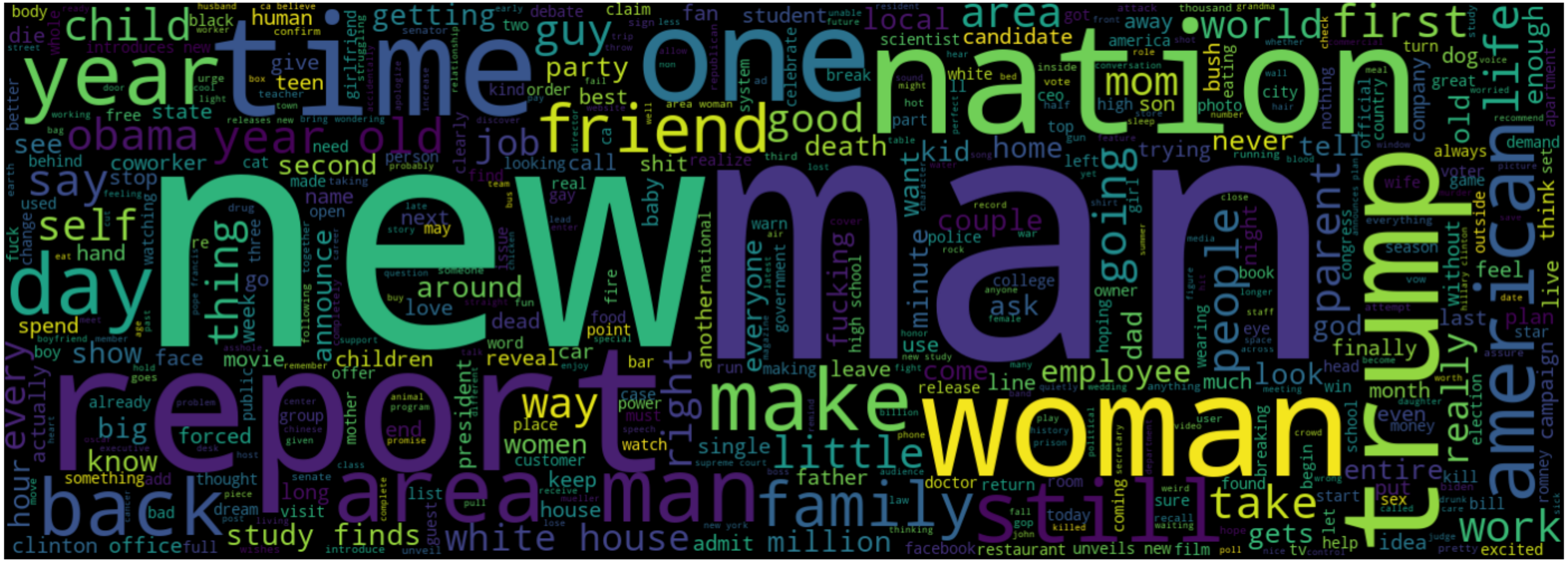
Trump and **Donald Trump** appear to be the most common words. **New** and **Say** are also common.



Word Cloud for non-sarcastic headlines

DATASET VISUALIZATION (2)

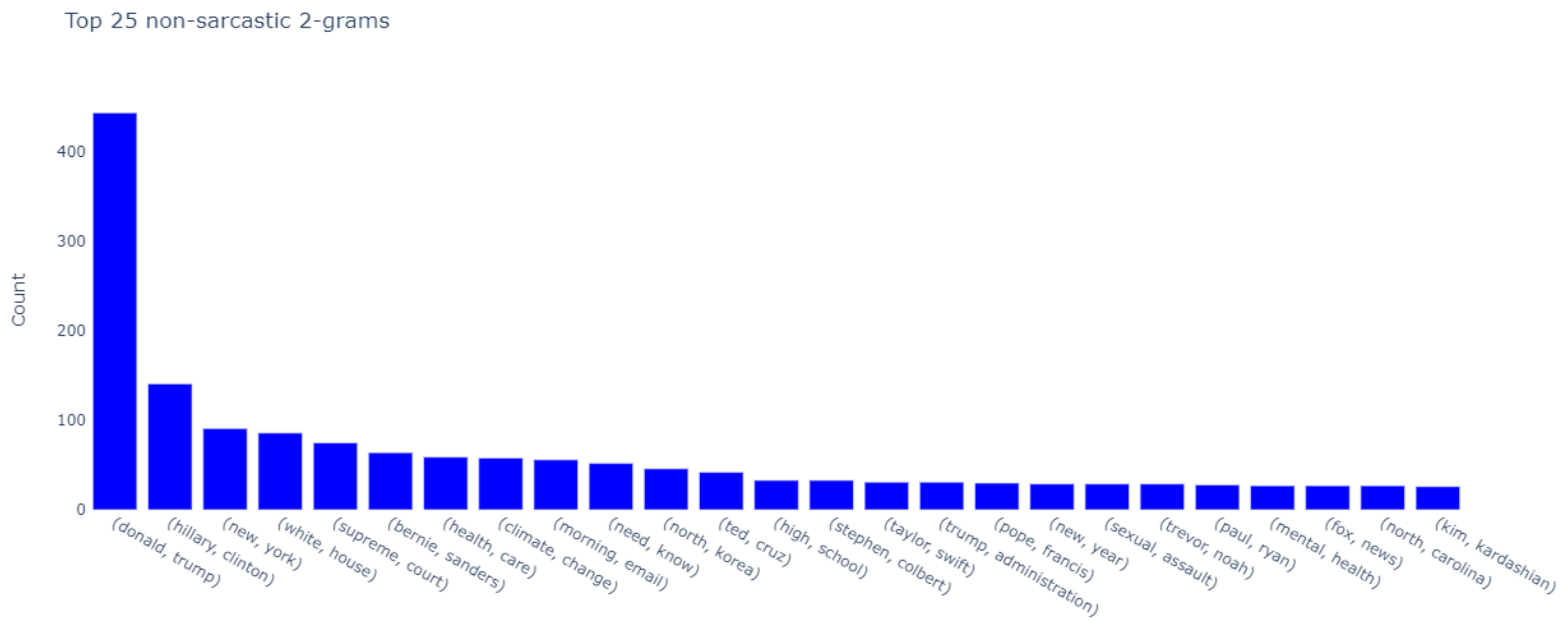
New is in common with the non-sarcastic headlines. **Man** and **Report** appear to be the most common words.



Word Cloud for sarcastic headlines

DATASET VISUALIZATION (3)

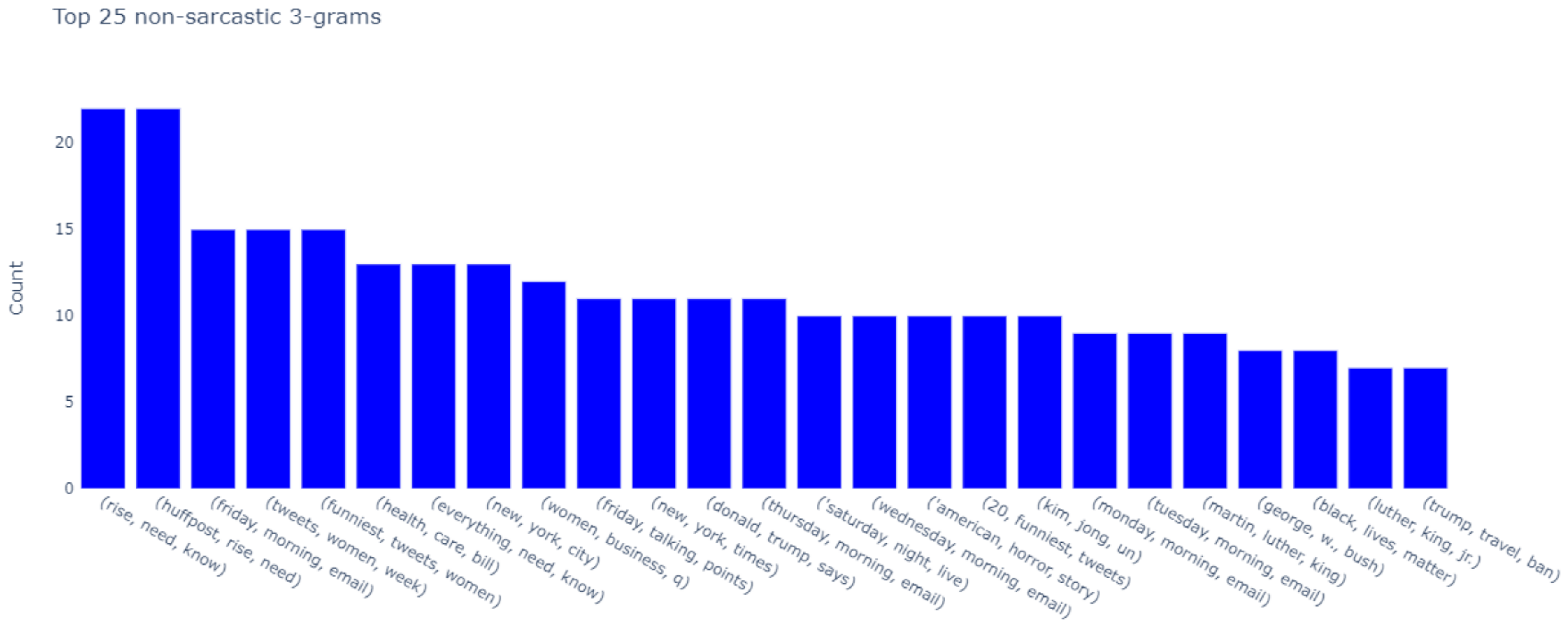
The most common 2-gram is **Donald Trump**, followed by **Hilary Clinton**.



Most common 2-grams for non-sarcastic headlines

DATASET VISUALIZATION (4)

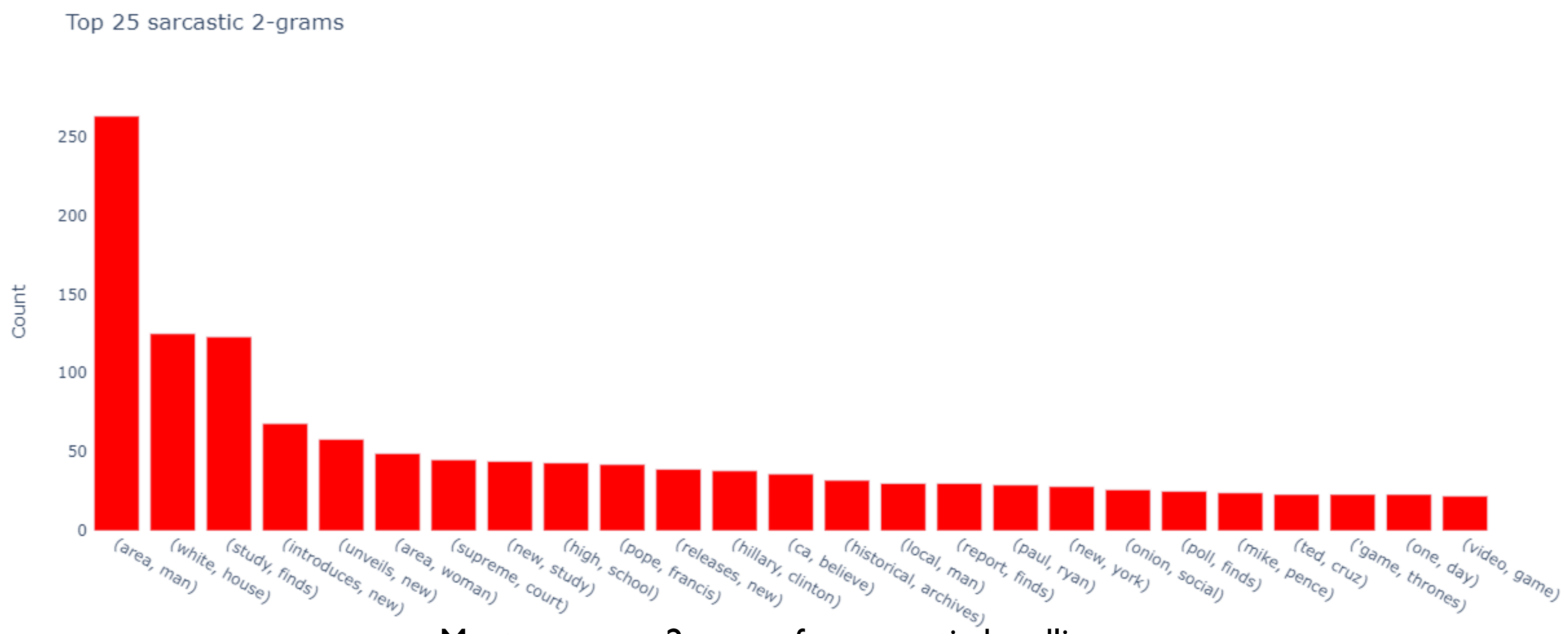
The two most common 3-grams come from the HuffPost daily recaps, such as *HuffPost Rise: What You Need To Know On August 12.*



Most common 3-grams for non-sarcastic headlines

DATASET VISUALIZATION (5)

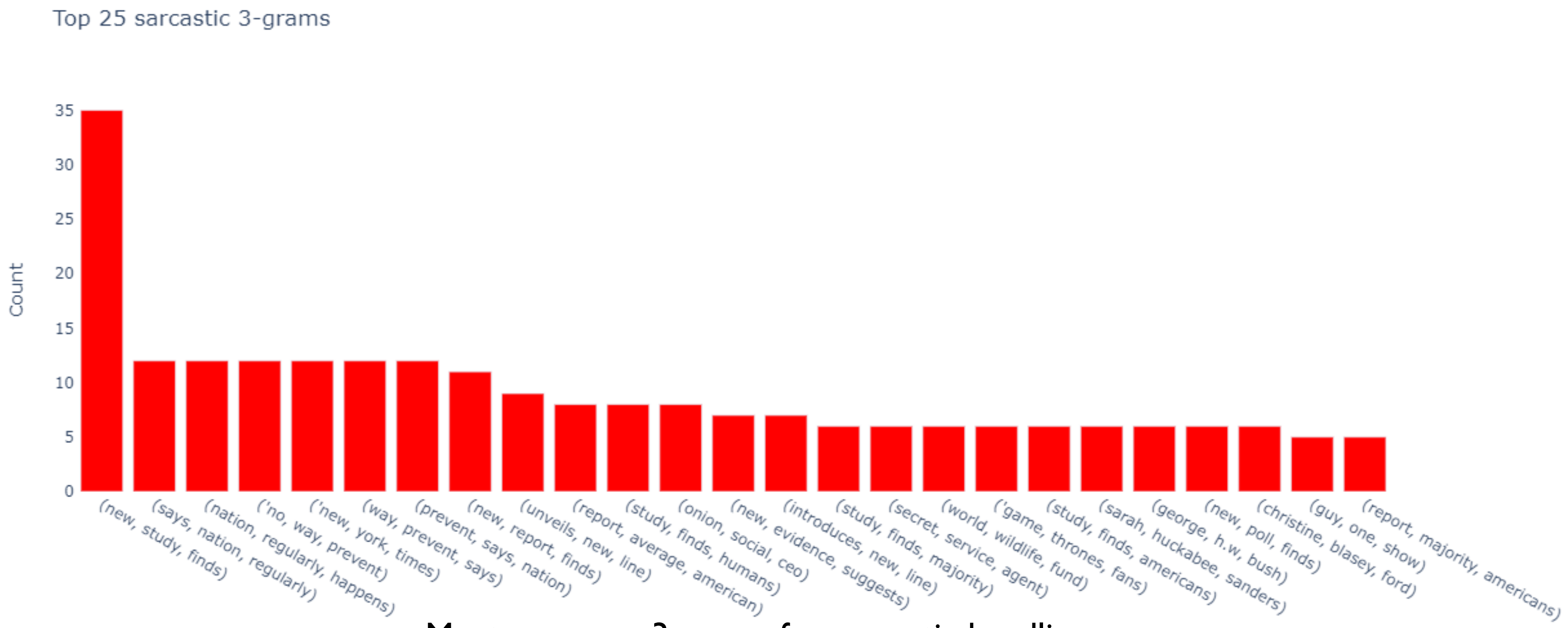
The most 2-grams relates to The Onion's common joke headlines referring to area/local men reporting something, or studies that find something new. There are also political bigrams such as **White House**.



Most common 2-grams for sarcastic headlines

DATASET VISUALIZATION (6)

The first 3-gram refers to new studies finding some things. The following ones all refer to the Onion's different series of satirical articles such as *No Way To Prevent This,' Says Only Nation Where This Regularly Happens*.

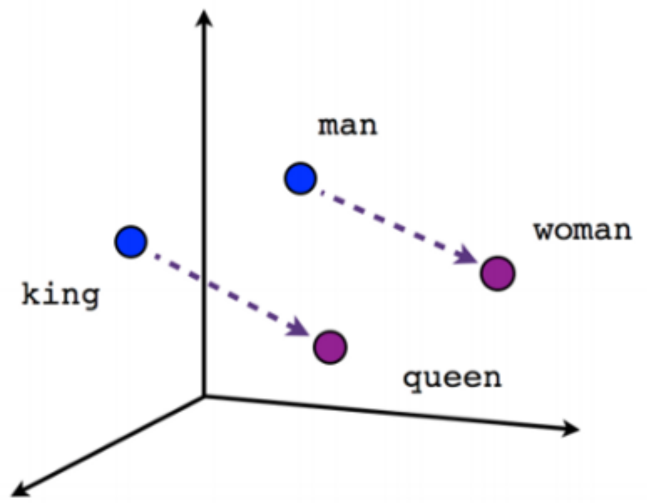


Most common 3-grams for sarcastic headlines

WORD EMBEDDINGS – AN INTRODUCTION

- Natural language is a **high dimensional** and **multi-sense** problem domain dealing with polysemy, synonymy, antonymy and hyponymy [10]
- In order to detect sarcasm effectively, we need to:
  - Consider models that (usually) work with "**windows**" that encompass multiple words at the same time
  - **Reduce the dimensionality** of the inputs so that a machine can handle them, i.e. map the words into a plane

Word Embeddings exploit the **Distributional Hypothesis** [11][12] of language, which states that the meaning of a word is a function of the contexts in which it appears



PRE-TRAINED WORD EMBEDDINGS

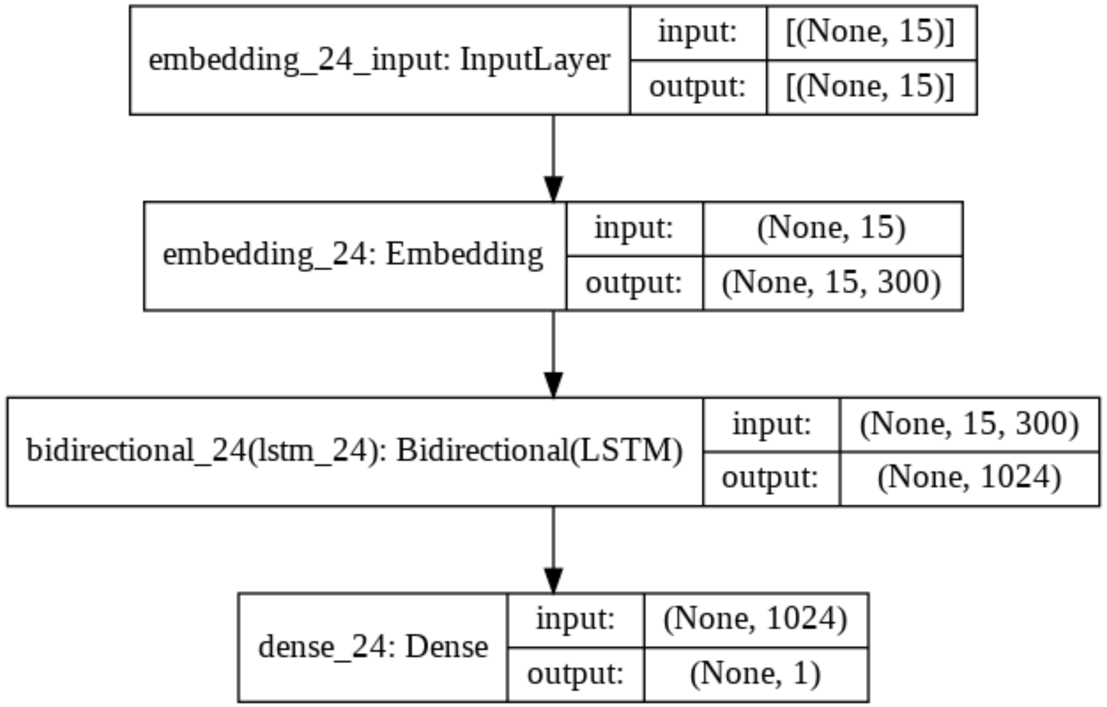
- **Word2Vec** was first introduced by Mikolov et al. (2013) [13] and has become the *de facto standard* in a lot of language downstream tasks due to its **efficiency** and **quality** of the generated representations
- **fastText** [14] is a language model inspired by Word2Vec. Instead of learning vectors for words directly, fastText represents each word as an n-gram of characters. This helps capture the meaning of shorter words and allows the embeddings to understand suffixes and prefixes
- **Global Vectors for Word Representation (GloVe)** [15] has no context window to obtain any local context. The main idea behind this model is that when two or more words occur multiple times, they may have a similar meaning

Word2Vec	• Trained on the Google News corpus (≃100B tokens)
fastText	• Trained on Wikipedia 2017, UMBC WebBase and statmt.org news dataset corpora (≃16B tokens)
Glove	• Trained on Wikipedia 2014 and Gigaword corpora (≃6B tokens)

In order to obtain (and work with) these pre-trained embeddings, we used the **Gensim** library (pypi.org/project/gensim/)

NEURAL NETWORK ARCHITECTURE

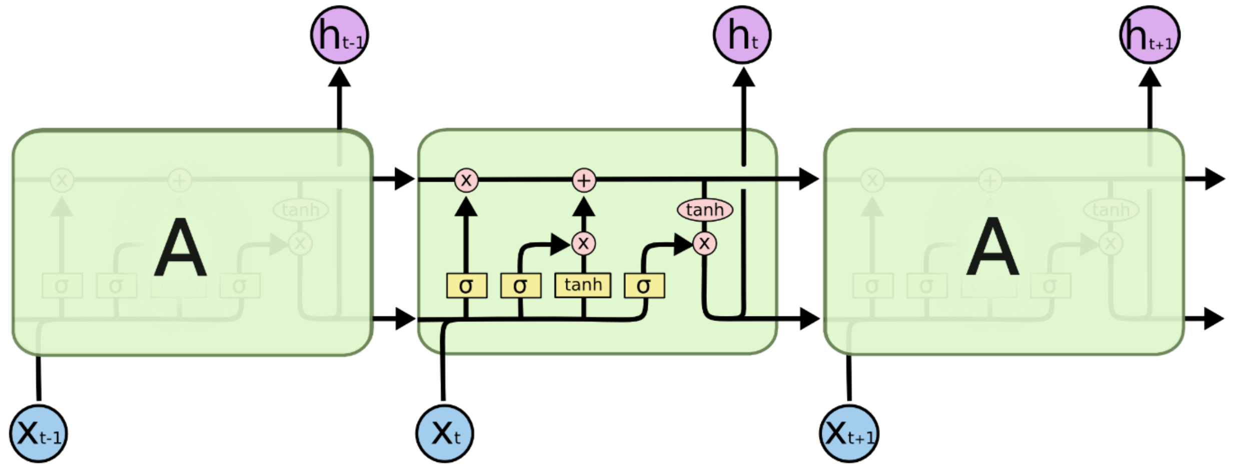
- The proposed **Deep Learning architecture** has the following layers:
  - **Embedding** layer
  - **Bidirectional LSTM** layer [16]
  - **Output** layer
- By taking inspiration from **CADE** [17], we "*freeze*" the Embedding layer and use the weights obtained from the three Word Embedding models
- The inputs for the network are obtained as follows:
  1. We build a vocabulary with all the words in the headlines; each word is assigned an index based on the number of occurrencies
  2. We tokenize each headline, and assign to each token its respective index in the vocabulary



In the proposed architecture headlines are 15-words-long, while the embedding size is 300

RNNs AND BI-LTSM NETWORKS

- **Recurrent Neural Networks (RNNs)** [18] are types of neural networks that contain loops, allowing information to be stored within the network
  - They struggle with long-term dependencies
- A **Long Short Term Memory (LSTM)** [19] network is a special kind of RNN that is capable of remembering long-term relationships while also being able to forget unnecessary information
- We use a **Bidirectional LSTM**, where two LSTM networks are trained at the same time: one on the input as-is, and the other on the reversed input

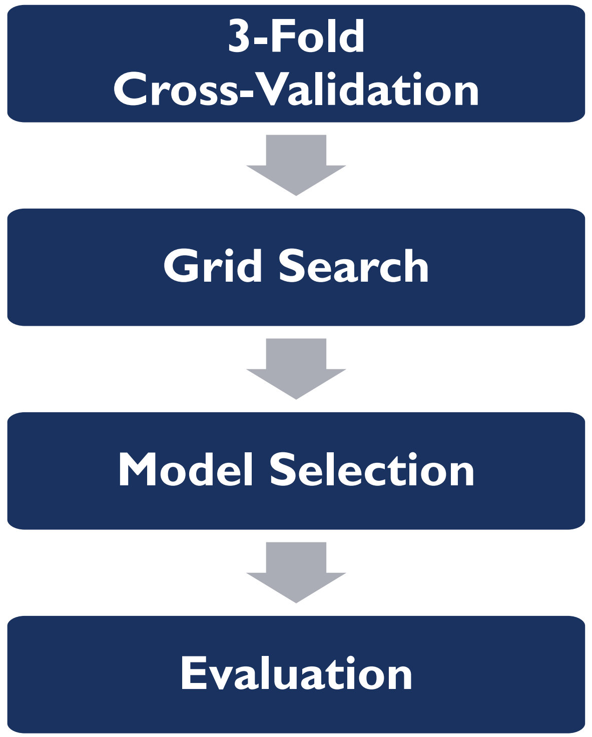


For the classification task, we used the **Keras** (keras.io/) and the **scikit-learn** (scikit-learn.org/stable/) libraries

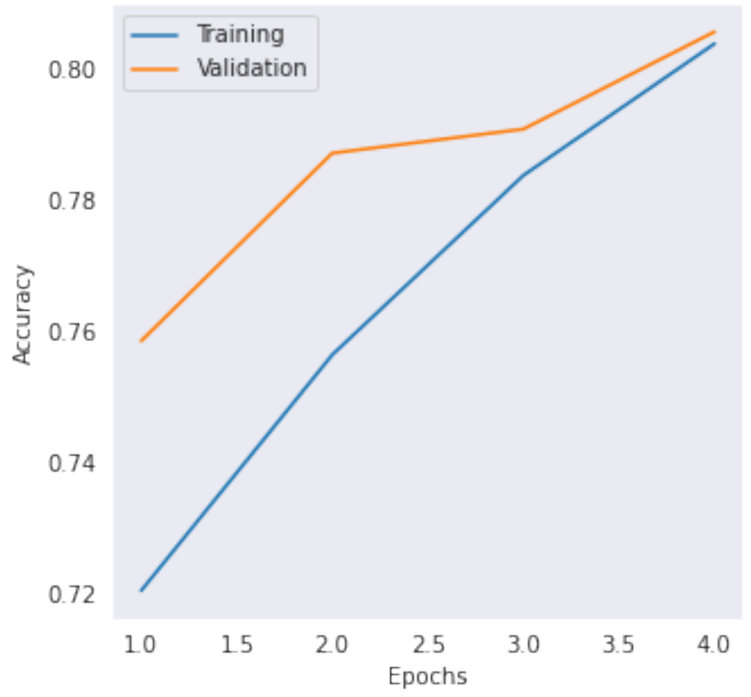
METHODOLOGY

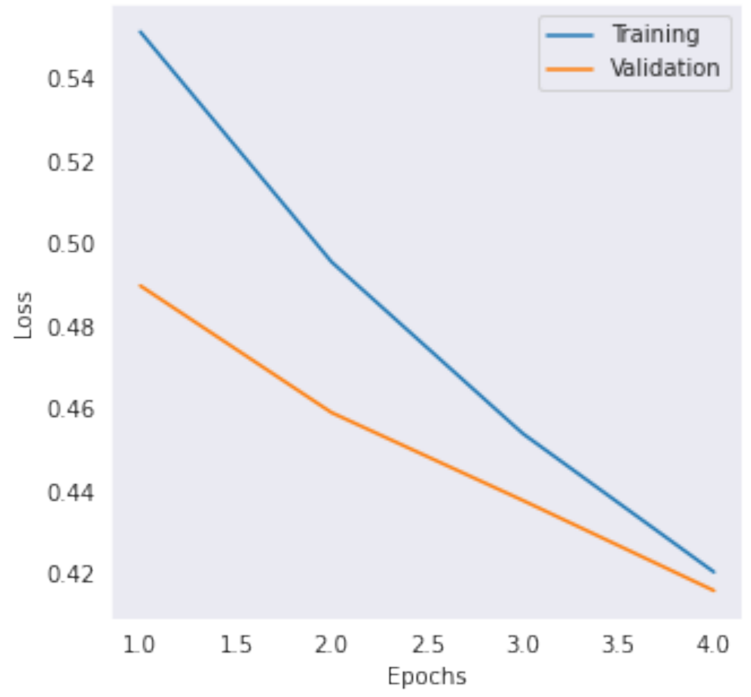
- 3-fold cross-validation on the training data via *grid search* (8 experiments in total)
- Average validation accuracy to perform **model selection**
- Split fine-tuning data into 80% train and 20% validation to further train and detect *overfitting*
- Evaluation on test data

Hyperparameters: *epochs*, *batch_size*, *units*, *dropout rate*



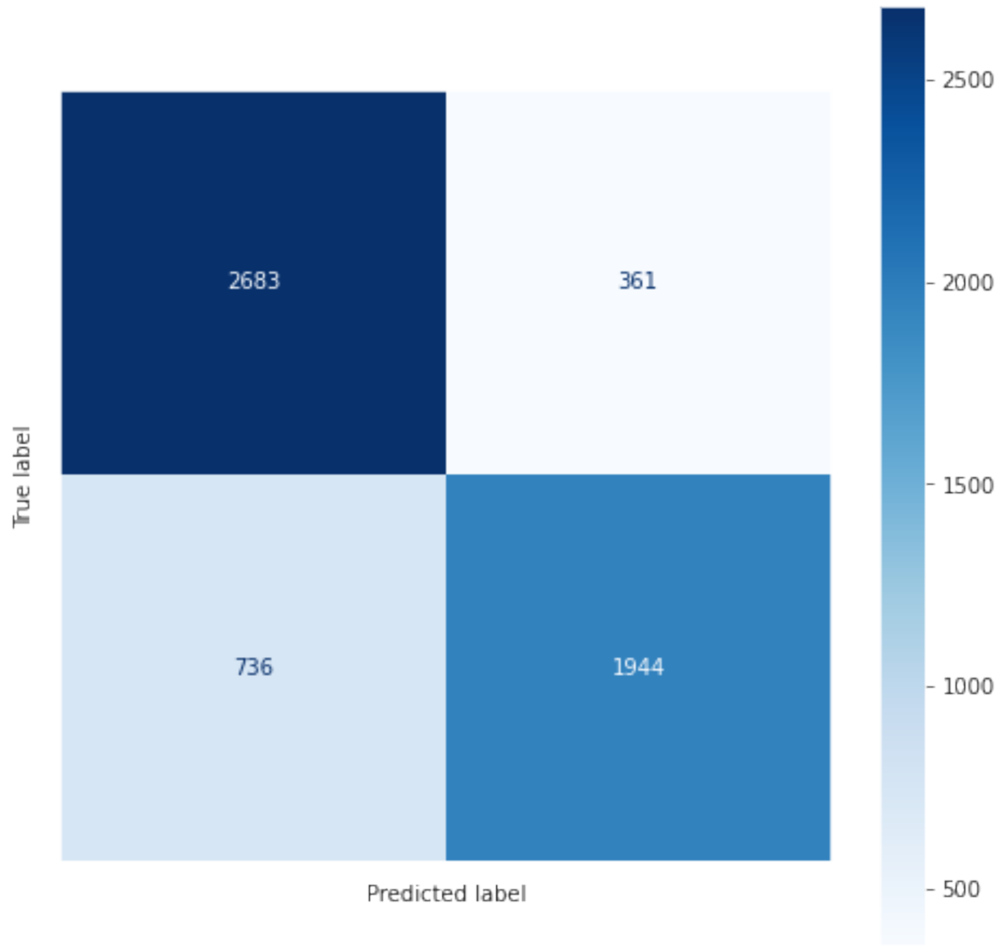
WORD2VEC – ACCURACY AND LOSS





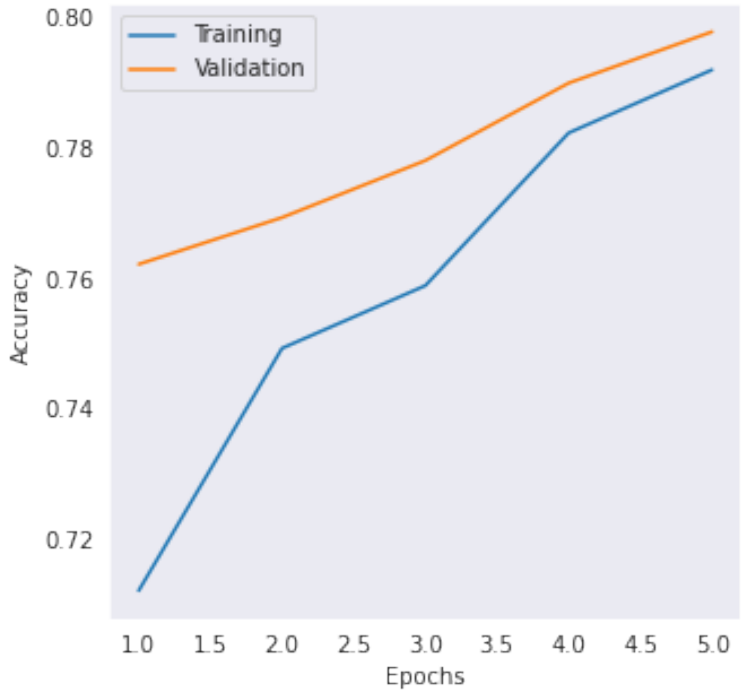
This model does not show overfitting, which isn't always the case with textual data.
This result was achieved with a dropout of 0.5 and a batch size of 32.
By analyzing the predictions on the test set, we reached an accuracy of 80.83%.

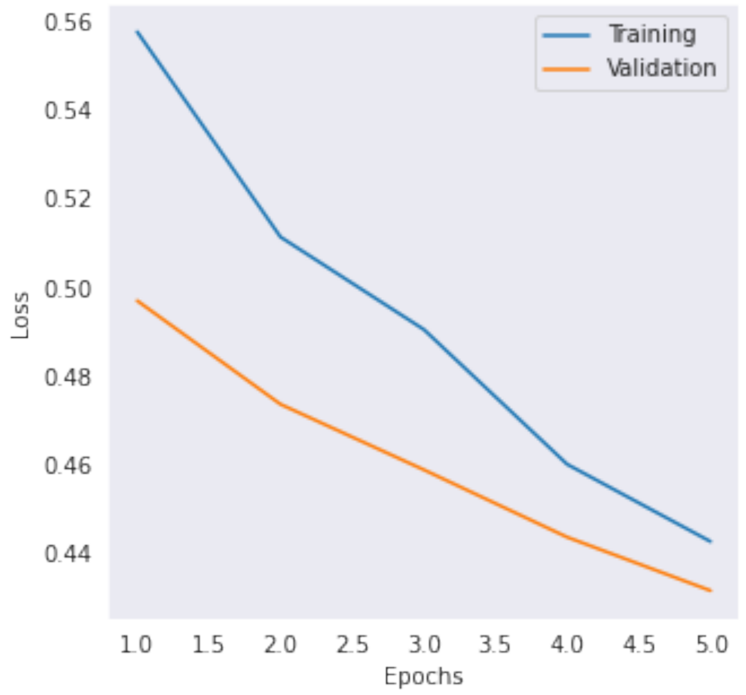
WORD2VEC – CONFUSION MATRIX



From the confusion matrix, we can see that the number of false negatives is more than double the number of false positives, this means that the model classifies with a greater probability non-sarcastic headlines as sarcastic ones

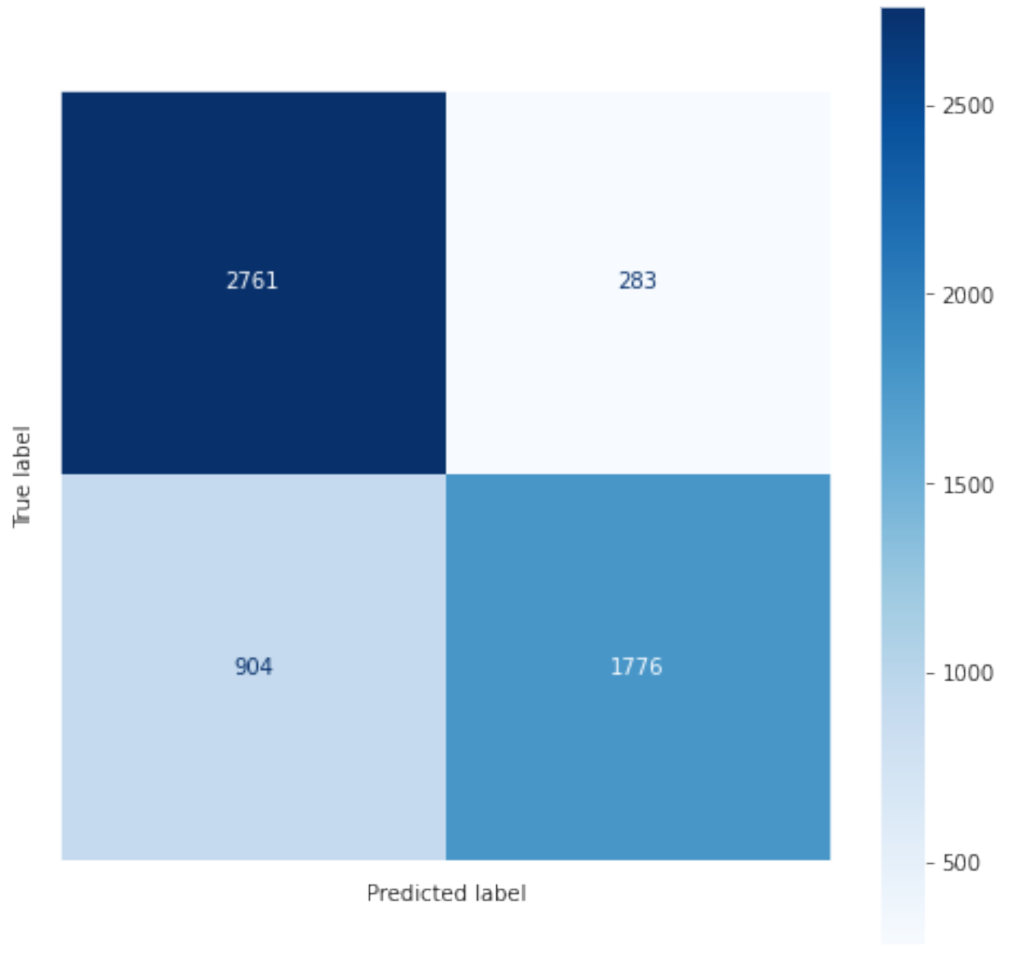
FASTTEXT – ACCURACY AND LOSS





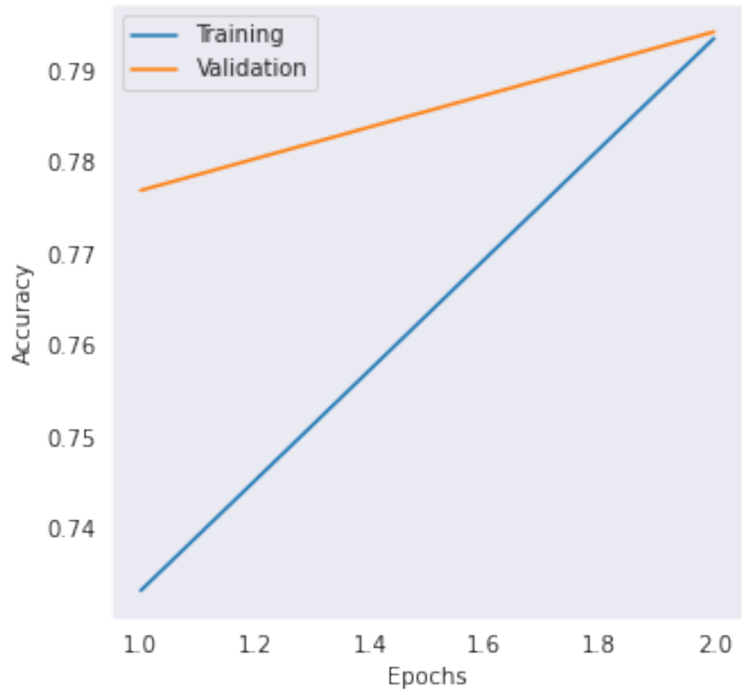
This model does not show overfitting, while also increasing the number of epochs to 5.
In addition, this result was achieved with a dropout of 0.5 and a batch size of 32.
By analyzing the predictions on the test set, we reached an accuracy of 79.55%.

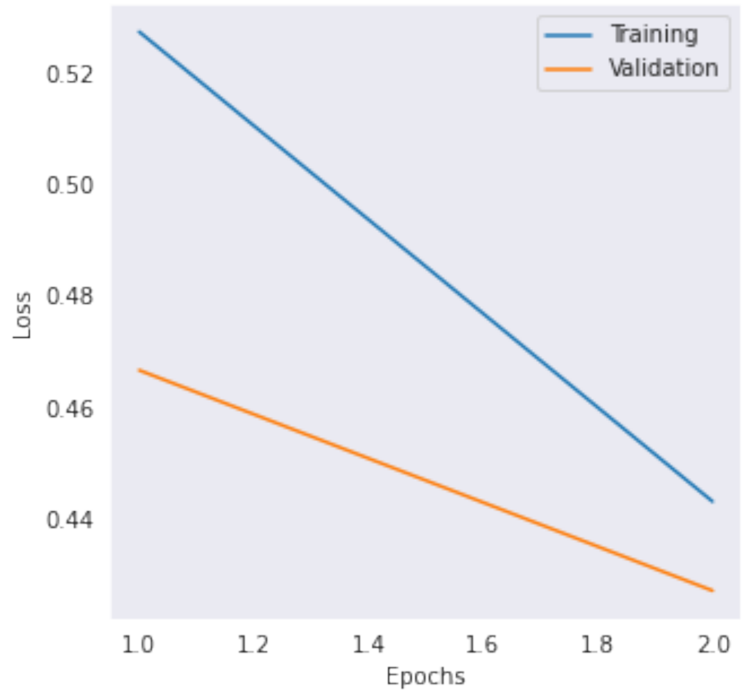
FASTTEXT – CONFUSION MATRIX



From the confusion matrix, we can see that the number of false negatives is much greater that the number of false positives

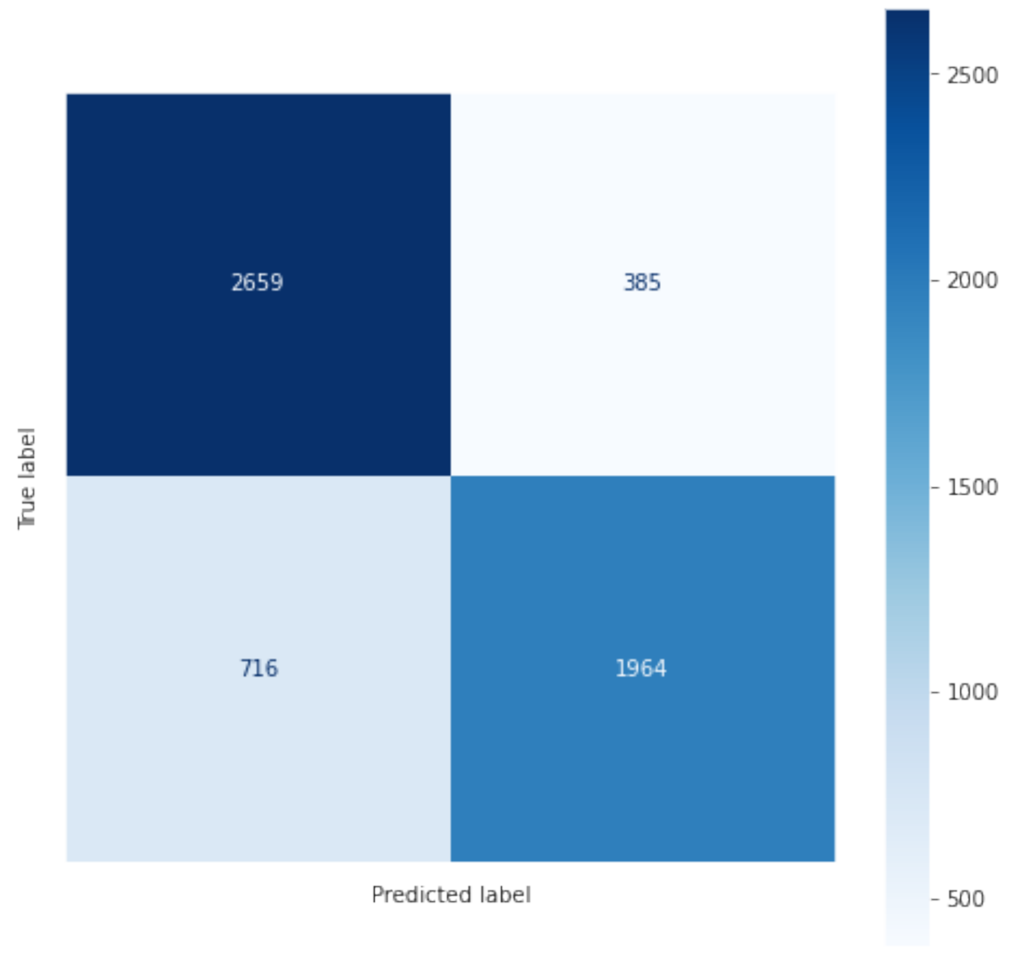
GLOVE – ACCURACY AND LOSS





This model does not show overfitting, but initially shows very far values for training and validation in terms of accuracy and loss.
We notice that the number of epochs is low (only 2) and a dropout of 0.4.
By analyzing the predictions on the test set, we reached an accuracy of 80.50%.

GLOVE – CONFUSION MATRIX



From the confusion matrix, we can see that the number of false negatives is almost double the number of false positives, similarly to Word2Vec

RECAP AND COMPARISON

Word Embeddings	Proposed	State-of-the-art (LSTM-CNN)
Word2Vec	80.83%	81.23%
fastText	79.55%	81.45%
Glove	80.50%	**81.60%**

Regarding the proposed architecture, the best model in terms of accuracy was Word2Vec with an accuracy of 80.83%.

When compared to the state-of-the-art [20], which reaches 81.60% by using Glove, we reached comparable results, even though <u>our architecture is simpler</u> that the one presented in the literature.

SEMANTIC ANALYSIS – COMPARING WORD EMBEDDINGS

- Comparing the (pre-trained) Word Embedding that we used is not straightforward:
  - Different **models**
  - Different training **corpora**
- Training Word Embeddings on real data yields **bias**
  - This bias <u>may propagate</u> in all NLP tasks that use the Embeddings
- Even though training corpora may be very large, some words may still be missing:
  - **Out Of Vocabulary** is a real issue

We can do an *intrinsic comparison:* given some key-entities, check if they share similar neighbors

SEMANTIC ANALYSIS – OUT OF VOCABULARY

- Since we use *pre-trained* Word Embeddings, some words in some models may be missing, for example:
  - *Abbreviations and Numbers (e.g. snl, 2015, …)*
  - *Celebrities (e.g. Ivanka, Weinstein, Beyonce, …)*
  - *Specific lexicon (e.g. huffpolster, …)*
- Does this impact Sarcasm Detection classification? According to the accuracies, **this does not seem to be the case**

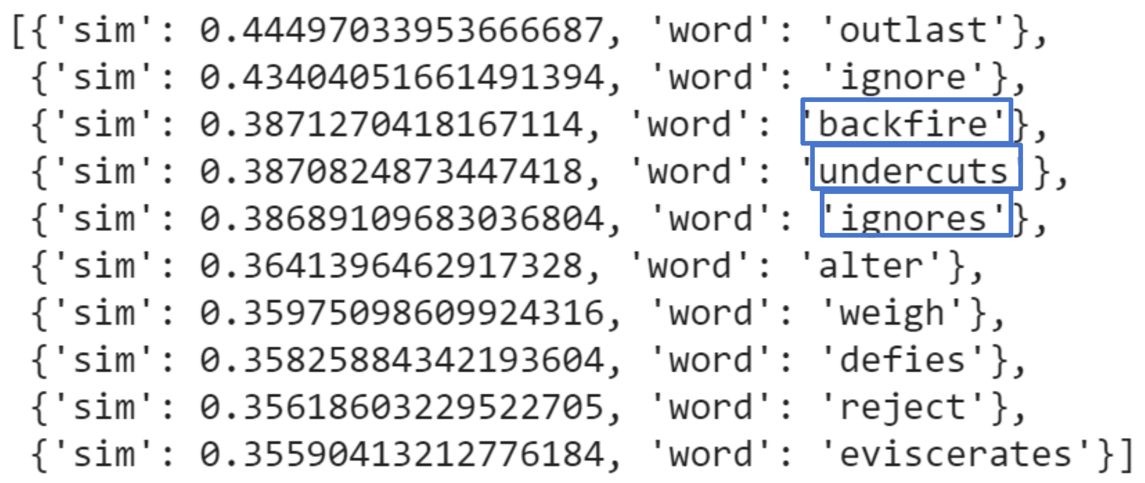
Word Embeddings	OOV - Absolute	OOV - Relative
Word2Vec	5606	18%
fastText	4214	14%
Glove	2886	9%

SEMANTIC ANALYSIS – INTRINSIC COMPARISON

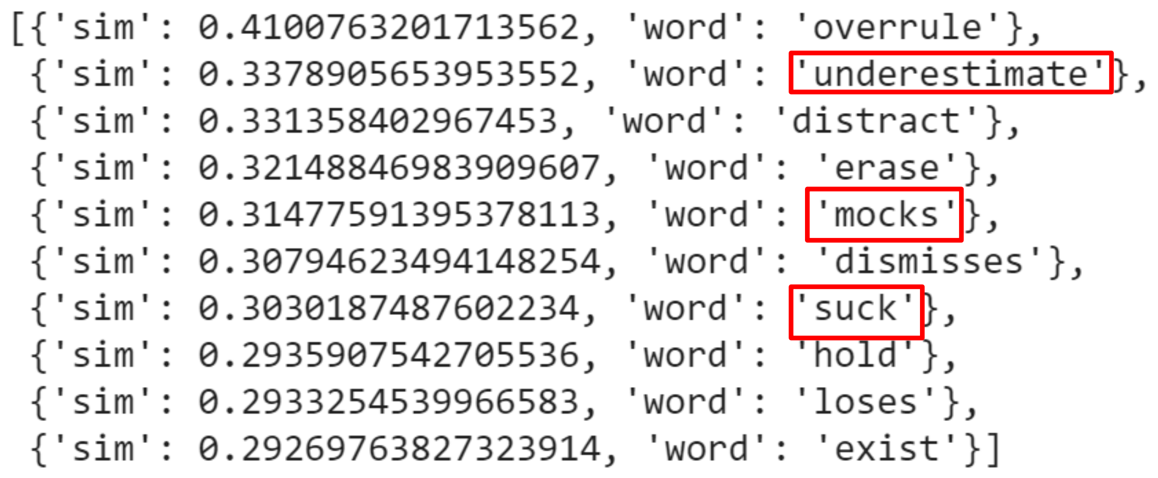
- We focused on the **Trump** entity (without considering ambiguities)
  - Highly controversial and divisive
  - Most common unigram in both sarcastic and non-sarcastic headlines
- First, we obtain all headlines that contain the word **Trump**
  - We split these headlines into *sarcastic* and *non-sarcastic* ones
- Then, we obtain the set of words that appear in these headlines, and we compute similarities:
  - Which entities does each Word Embedding consider more similar to **Trump**?
  - Are these consistent across different Word Embeddings?
  - Are specific entities more related to sarcasm than others?

SEMANTIC ANALYSIS – WORD2VEC

- In sarcastic headlines, **Trump** is mostly related to aggressive verbs, perhaps pointing to his highly egocentric character. In non-sarcastic headlines, **Trump** is closer to "negative" verbs.



```
[{'sim': 0.4449703395366687, 'word': 'outlast'},
 {'sim': 0.43404051661491394, 'word': 'ignore'},
 {'sim': 0.3871270418167114, 'word': 'backfire'},
 {'sim': 0.3870824873447418, 'word': 'undercuts'},
 {'sim': 0.38689109683036804, 'word': 'ignores'},
 {'sim': 0.3641396462917328, 'word': 'alter'},
 {'sim': 0.35975098609924316, 'word': 'weigh'},
 {'sim': 0.35825884342193604, 'word': 'defies'},
 {'sim': 0.35618603229522705, 'word': 'reject'},
 {'sim': 0.35590413212776184, 'word': 'eviscerates'}]
```

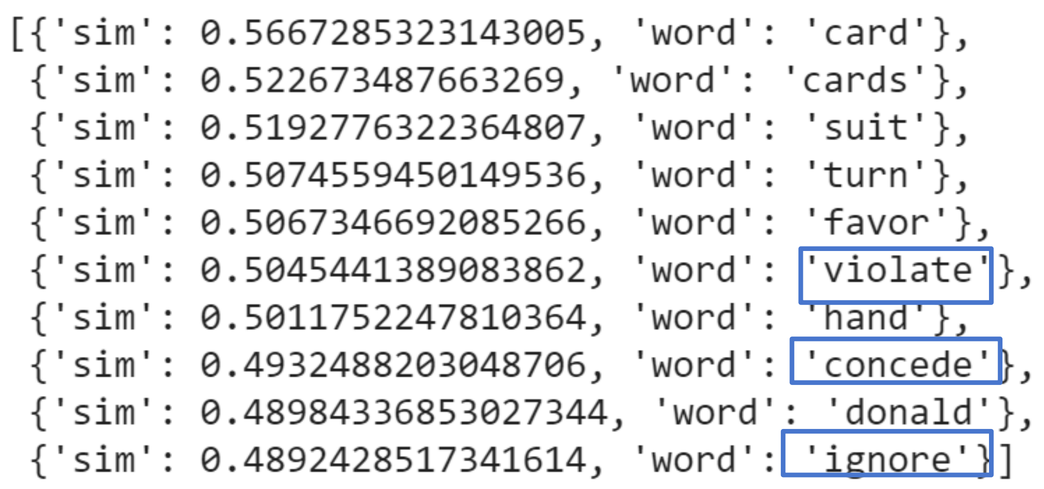


```
[{'sim': 0.4100763201713562, 'word': 'overrule'},
 {'sim': 0.3378905653953552, 'word': 'underestimate'},
 {'sim': 0.331358402967453, 'word': 'distract'},
 {'sim': 0.32148846983909607, 'word': 'erase'},
 {'sim': 0.31477591395378113, 'word': 'mocks'},
 {'sim': 0.30794623494148254, 'word': 'dismisses'},
 {'sim': 0.3030187487602234, 'word': 'suck'},
 {'sim': 0.2935907542705536, 'word': 'hold'},
 {'sim': 0.2933254539966583, 'word': 'loses'},
 {'sim': 0.29269763827323914, 'word': 'exist'}]
```

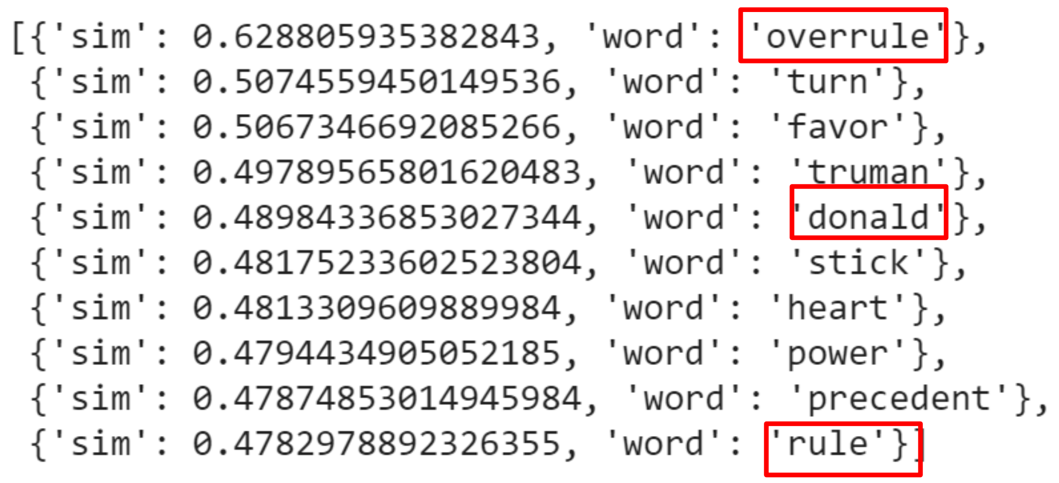
Top 10 most similar words to " Trump" in non-sarcastic and sarcastic headlines

SEMANTIC ANALYSIS – FASTTEXT

- fastText's similarities are harder to interpret in a meaningful way.



```
[{'sim': 0.5667285323143005, 'word': 'card'},
 {'sim': 0.522673487663269, 'word': 'cards'},
 {'sim': 0.5192776322364807, 'word': 'suit'},
 {'sim': 0.5074559450149536, 'word': 'turn'},
 {'sim': 0.5067346692085266, 'word': 'favor'},
 {'sim': 0.5045441389083862, 'word': 'violate'},
 {'sim': 0.5011752247810364, 'word': 'hand'},
 {'sim': 0.4932488203048706, 'word': 'concede'},
 {'sim': 0.48984336853027344, 'word': 'donald'},
 {'sim': 0.4892428517341614, 'word': 'ignore'}]
```

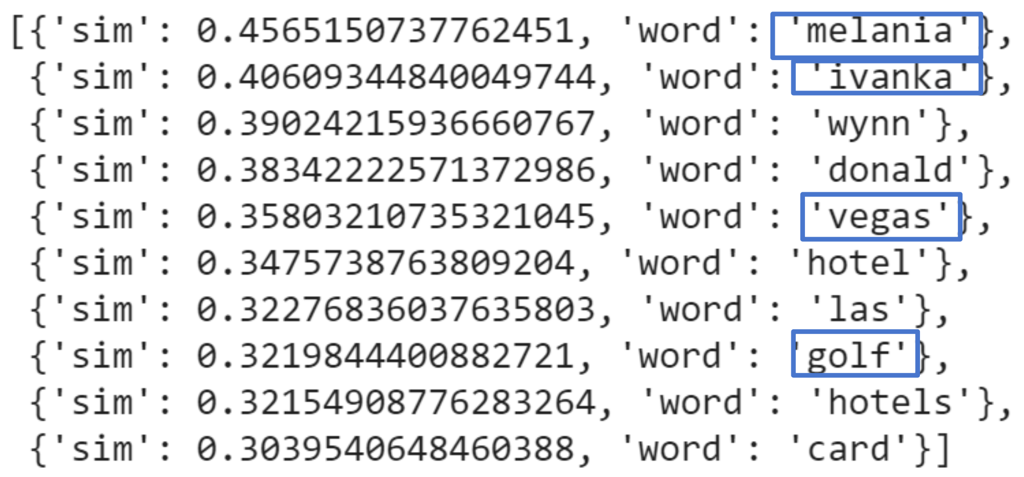


```
[{'sim': 0.628805935382843, 'word': 'overrule'},
 {'sim': 0.5074559450149536, 'word': 'turn'},
 {'sim': 0.5067346692085266, 'word': 'favor'},
 {'sim': 0.49789565801620483, 'word': 'truman'},
 {'sim': 0.48984336853027344, 'word': 'donald'},
 {'sim': 0.48175233602523804, 'word': 'stick'},
 {'sim': 0.4813309609889984, 'word': 'heart'},
 {'sim': 0.4794434905052185, 'word': 'power'},
 {'sim': 0.47874853014945984, 'word': 'precedent'},
 {'sim': 0.4782978892326355, 'word': 'rule'}]
```

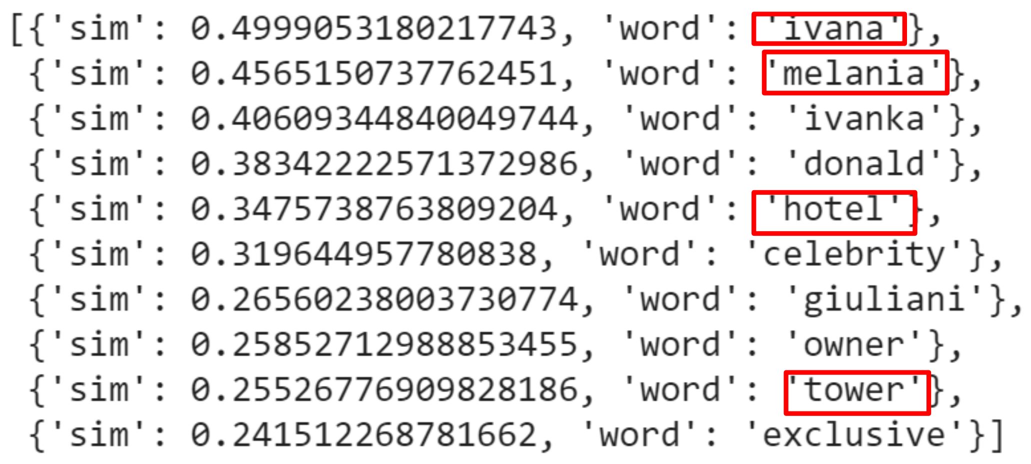
Top 10 most similar words to "Trump" in non-sarcastic and sarcastic headlines

SEMANTIC ANALYSIS – GLOVE

- Glove returns entities related to Trump's private life (such as **Melania** and **Ivanka**) and related businesses.



```
[{'sim': 0.4565150737762451, 'word': 'melania'},
 {'sim': 0.40609344840049744, 'word': 'ivanka'},
 {'sim': 0.3902421593666076, 'word': 'wynn'},
 {'sim': 0.3834222571372986, 'word': 'donald'},
 {'sim': 0.35803210735321045, 'word': 'vegas'},
 {'sim': 0.3475738763809204, 'word': 'hotel'},
 {'sim': 0.32276836037635803, 'word': 'las'},
 {'sim': 0.3219844400882721, 'word': 'golf'},
 {'sim': 0.32154908776283264, 'word': 'hotels'},
 {'sim': 0.3039540648460388, 'word': 'card'}]
```



```
[{'sim': 0.4999053180217743, 'word': 'ivana'},
 {'sim': 0.4565150737762451, 'word': 'melania'},
 {'sim': 0.40609344840049744, 'word': 'ivanka'},
 {'sim': 0.3834222571372986, 'word': 'donald'},
 {'sim': 0.3475738763809204, 'word': 'hotel'},
 {'sim': 0.319644957780838, 'word': 'celebrity'},
 {'sim': 0.26560238003730774, 'word': 'giuliani'},
 {'sim': 0.2585271298853455, 'word': 'owner'},
 {'sim': 0.25526776909828186, 'word': 'tower'},
 {'sim': 0.241512268781662, 'word': 'exclusive'}]
```

Top 10 most similar words to "Trump" in non-sarcastic and sarcastic headlines

CONCLUSION

- We have presented our architecture for Sarcasm Detection with a "frozen" Embedding layer that utilizes the weights obtained by a pre-trained Word Embedding. We compared Word2Vec, fastText and Glove and reached an accuracy of **80.83%** on the *News Headlines Dataset for Sarcasm Detection* dataset. We also avoid overfitting
- Our architecture is comparable with current state-of-the-art approaches, while also being *simpler,* due to it not using hybrid networks
- By analyzing a more cognitive dimension, we tried to analyze the knowledge of different Word Embeddings about sarcasm on the Donald Trump entity. Trump was chosen due to it being the most common entity on *both* sarcastic and non-sarcastic headlines. We found out that each Word Embedding prioritized different concepts in both sarcastic and non-sarcastic headlines

FUTURE WORKS

- Make the architecture more complex (e.g. by adding a **CNN layer**)
- Test our model with other datasets (such as social networks data)
- Use Word Embeddings as initialization weights, rather than “freezing” the layer altogether
- Make a more in-depth analysis on the semantic aspects, by using specific tools to deal with **ambiguities** and other related problems

REFERENCES

[1] Joshi, A., Sharma, V., & Bhattacharyya, P. (2015). Harnessing Context Incongruity for Sarcasm Detection. Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), 757–762.

[2] Joshi, A., Bhattacharyya, P., & Carman, M. J. (2017). Automatic sarcasm detection: A survey. ACM Computing Surveys (CSUR), 50(5), 1-22.

[3] Gonzalez-Ibanez, R., Muresan, S., & Wacholder, N. (2011). Identifying sarcasm in twitter: A closer look. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers, 2, 581–586. Association for Computational Linguistics.

[4] Maynard, D., & Greenwood, M.A. (2014). Who cares about sarcastic tweets? Investigating the impact of sarcasm on sentiment analysis. In Proceedings of LREC 2014. ELRA.

[5] Liebrecht, C., Kunneman, F., & van Den Bosch, A. (2013). The perfect solution for detecting sarcasm in tweets #not. Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA).

[6] Pozzi, F.A., Fersini, E., Messina, E., Liu, B.: Sentiment Analysis in Social Networks. Morgan Kaufmann, Burlington, MA (2016)

[7] Felbo, B., Mislove, A., Søgaard, A., Rahwan, I., & Lehmann, S. (2017). Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. arXiv preprint arXiv:1708.00524

[8] Misra, R., & Arora, P. (2019). Sarcasm detection using hybrid neural network. arXiv preprint arXiv:1908.07414.

[9] Krouska, A., Troussas, C., & Virvou, M. (2016, July). The effect of preprocessing techniques on Twitter sentiment analysis. In 2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA) (pp. 1-5). IEEE.

[10] Nitsche, M., & Tropmann-Frick, M. (2018, October). Context and Embeddings in Language Modelling-an Exploration. In DAMDID/RCDL (pp. 131-138).

[11] Harris, Z. S. (1954). Distributional structure. Word, 10(2-3), 146-162.

[12] Firth, J. R. (1957). A synopsis of linguistic theory, 1930-1955. Studies in linguistic analysis.

[13] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. Workshop on ICLR.

[14] Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, 5, 135–146.

[15] Pennington, J., Socher, R., & Manning, C.D. (2014). Glove: Global vectors for word representation. In Proceedings of EMNLP, 1532–1543. ACL.

[16] Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. IEEE transactions on Signal Processing, 45(11), 2673-2681.

[17] Bianchi, F., Di Carlo, V., Nicoli, P., & Palmonari, M. (2020). Compass-aligned Distributional Embeddings for Studying Semantic Differences across Corpora. arXiv preprint arXiv:2004.06519.

[18] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). Learning internal representations by error propagation. California Univ San Diego La Jolla Inst for Cognitive Science.

[19] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.

[20] Mehndiratta, P., & Soni, D. (2019). Identification of sarcasm in textual data: A comparative study. Journal of Data and Information Science, 4(4), 56-83.