PREDICTING FAKE NEWS

DATA SCIENCE 2022

Gabrielle H. B. Madsen	**Jakob H. Schauser**	**Martin Pries-Brøndberg**	**Monika Haubro**
lxs789	pwn274	cmw525	qhg541
Task 2 and 3, 50% *Task 5, 25%*	*Task 1 and 4, 50%* *Task 5, 25%*	*Task 2 and 3, 50%* *Task 5, 25%*	*Task 1 and 4, 50%* *Task 5, 25%*

June 9, 2022

1 Know your data

1 In order to represent the FakeNewsCorpus (FNC) dataset (1,000,000 rows), we created a database in pgAdmin4, 'FakeNews', and started removing redundant columns. Columns were deemed redundant if they contained zero entries or a single entry. Furthermore, the column `scraped_at` was removed as only two entries existed, and the given times were four hours apart. The resulting table was 'Million' (see appendix 6), where we also noted that our sample had an overerepresentation of articles with `type = fake` compared to the full dataset.

As we assumed we would have to normalize it later, we chose to create four dimension tables, 'dim_authors', 'dim_domain', 'dim_type' and 'dim_meta_keywords' (see appendix 6).
Based on the 'Million' and the dimension tables we created the 'fact_fake_news' table (see appendix 6) with ID-values taken from the four dimension tables.

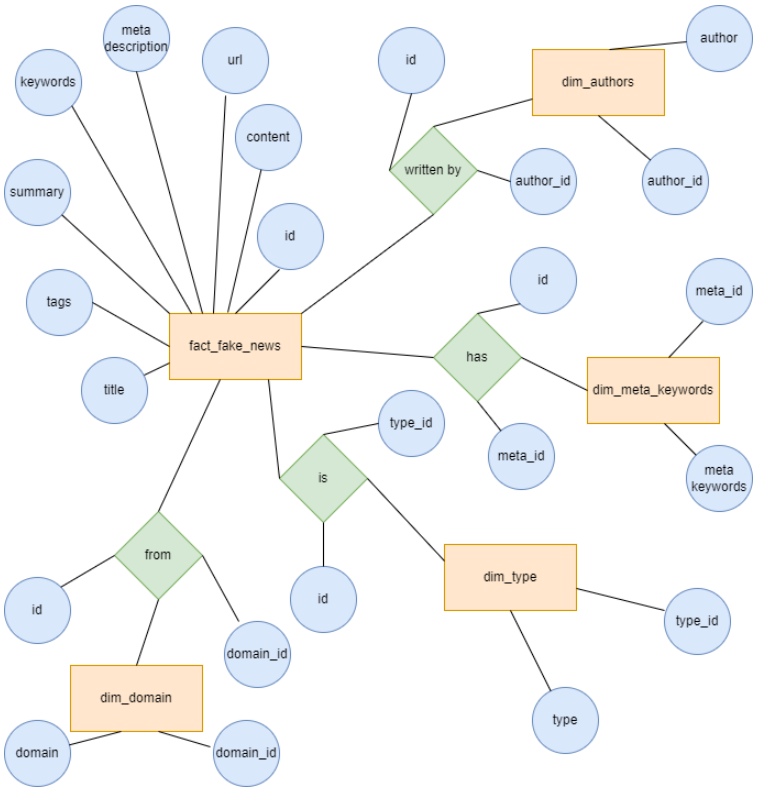
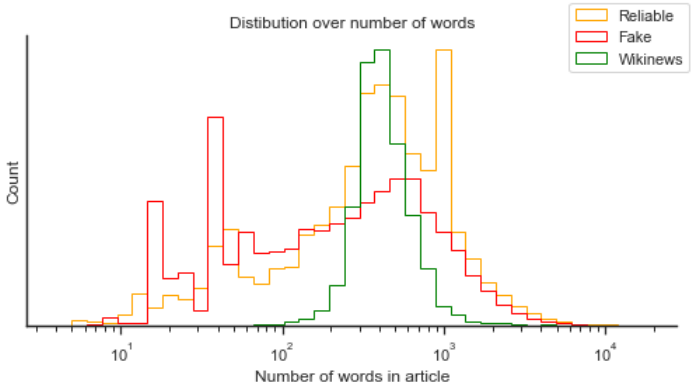


Figure 1: ER diagram of our database

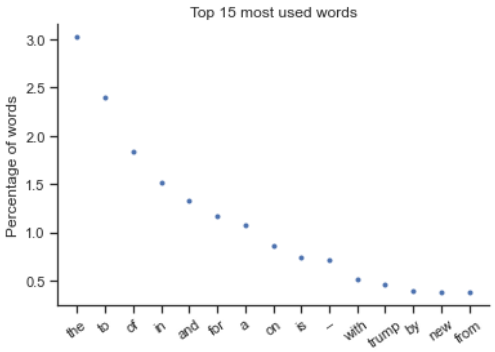
2 Before we loaded the 1,000,000 rows subset of the FakeNewsCorpus, we tried working with `news.csv.z01` and `news.csv.z02`, but encountered an error on around line 3586125 which we couldn't find a solution to. Aside from that, we mainly considered a problem in terms of the `type` labels, since the given type for any article was based on domain, implying the assumption that a domain only create a single type of articles which isn't necessarily true.

3 In terms of key properties of 'fact_fake_news', we plotted the distribution of words in the given articles with `type = fake`, `type = reliable` and compared them to the distribution of words in Wikinews articles.

We also plotted the top 15 most used words from 'fact_fake_news', without removing stopword, since we found it quite interesting that even with the presence of stopwords *trump* is still the 12th most common word.



(a) Distribution of amount of words on a log-scale for the fake, reliable and wikinews data.



(b) Top 15 most used words in the FNC

We note that both the `fake` and `reliable` articles from the FakeNewsCorpus has a very uneven distribution compared to the articles from Wikinews, which at a glance seems to follow a normal distribution. The `fake` articles also have as much larger presence in short articles compared to the `reliable` ones, which generally seem to be much longer.

4 After learning the how the website was constructed, extracting the data went relatively smoothly. The most difficult part were the cases where the articles were spread out over multiple result-pages, requiring multiple additional checks and queries. There were also a surprising lack of meta-information on many of the articles, making it difficult to extract something meaningful.

5 Since the extracted features only amounted to 'Title', 'Link', 'Date', 'Content' and 'Sources' there were not many choices to make in the design of the database. Since we didn't have any metadata available from Wikinews, we didn't have to consider it when solely looking at Wikinews. If we in Milestone 1 had also extractacted things like the url or the time of scraping, a more complicated database would have been necessary. But (as can be seen later) this would have been in vain, as we choose to not include the Wikinews data at all.

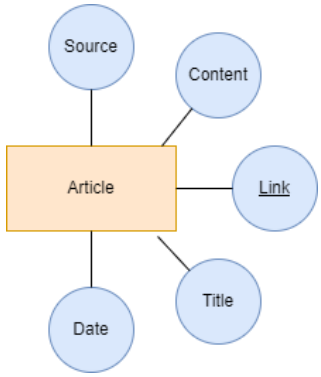
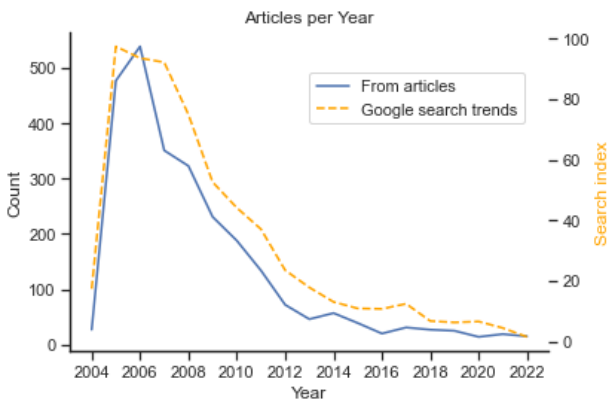


Figure 3: ER diagram for Wikinews

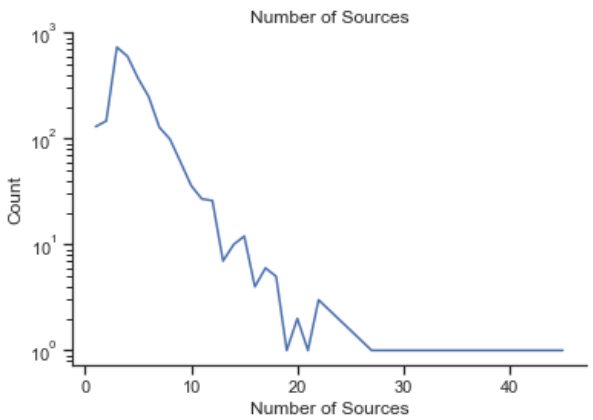
Article (Title:string, Link:string, Date:string, Content:string, Sources:string)

Table 1: Relational database schema for Wikinews.

6 We used SQL (see appendix 6) to find the basic statistics of Wikinews, and then we plotted our findings in figure 4a using python.
It seemed a bit odd that articles pr. year top in 2006, so we did a bit of research and found a graph of google searches for '*wikinews*' which we added to the plot. There we see a clear correlation between number of yearly searches for "Wikinews"[GoogleTrends] and articles per year in our scraped sample. We also plotted the amount of sources pr. article in figure 4b in order to get a better understanding of how external sources are used in Wikinews articles, since we in Milestone 1 became aware that anyone can edit Wikinews articles.



(a) Articles pr. year



(b) Number of sources

7 The following view was made for training:

```
CREATE VIEW modeling AS
    SELECT title, content, meta_description, meta_keywords FROM fake_news
    UNION
    SELECT title, content FROM wikinews
```

Since there were no common columns between the two databases except content and title (in which there were no overlaps), there were no way obvious way to map the two. Knowing that we will not be using the wikinews-data set for the ML alogrithm we simply chose to combine them in an join using all columns.

8 From the 'Million' dataset we created two datasets: One to train the machine learning algorithm (60%), and one to test it (40%).
As mentioned, we won't be including the wikinews data that we scraped in Milestone 1, since we concluded it wouldn't be reasonable to trust the articles. At the same time we don't believe that the articles are completely fake, which makes the scraped data ill-suited for a binary labeling of either FAKE or REAL.

FAKE	**REAL**
Fake	Reliable
Conspiracy	Political
Junksci	

9 For the 'Million' dataset we've set up the classification in terms of categories given in the rightside table.
Note that we've added more types than just those that are tagged "reliable" or "fake" in order to have more reasonable aomunt of articles to work with, as only 6601 articles are tagged "reliable" in the 1 million subset of FakeNews.
With our classification above we end up with more than 250k articels in both categories.

2 Establish a baseline

Baseline models are useful because they will function as benchmarks for our more complex models as baseline models set the minimum performance requirements. If a complex model is not outperforming these baseline models it is better to use a simpler model.

1 We have created four simple baselines for our Fake News predictor models:

The first baseline model, **Most Common Label (MCL)**, will look at the training set and counts how many fake and real labels there are and then adopt simply guess the type which is most frequent. Thus, the predictions of this model depends purely on the training set and do not consider any features from the article it will make a prediction on.
This model had an accuracy of 54.1%, but failed to predict any articles as fake news as it simply predicted all articles as real.

The second baseline model, **Guessing at random**, makes a random guess as to whether an article is fake or real. In other words, this model does not take any features into consideration before making predictions.
This model had an accuracy of 50.0%.

Choosing either of these models as our baseline is quite reasonable: If our more complex models cannot do any better, we might as well make a random guess.

We also created a **random forest** model, which simple to implement. We achieved an accuracy of 0.871.

2 The most prominent of the meta-data features are the meta-keywords. Since the meta-keywords is neither present in the Wikinews nor in the LIAR data-set we have chosen not to include these. To see the effect this has on the final prediction, we have tried training a model both with and without. We observed no improvement by any significance compared to the results when training without keywords.

3 Create a Fake News predictor

In the pursuit to create the best possible Fake News predictor we experimented with different models. We worked with three different types of models namely support vector machine (SVM), Naive Bayes (NB), and Long Short Term Memory (LSTM).

Selection of Models

We chose LSTM, since it works well on sequential data (Jose [2019]). With LSTM we could handle the content of the articles as sequences of words and train the model with these sequences and our labels. However it can take some time to train the model, and therefore we also included SVM and NB, which are faster to train. SVM is common in classification problems and can produce satisfactorily results using less power (GeeksForGeeks [2020]), while Naive Bayes is even more simple and fast to train and commonly used for text classification with success (JavatPoint).

Support Vector Machine (SVM)

Support Vector Machine (SVM) uses hyperplanes to classify and is "*One of the most efficient supervised learning methods for linear classification*" (Elliott, s. 27). For our SVM we use the sklearn Pipeline() to assemble transformers and an estimator in a certain order:

- CountVectorizer(), which converts the data into a count matrix
- TfidfTransformer(), which transform the count matrix to normalized term frequency
- SGDClassifier(), which is the SVM classifier or the estimator

We call fit() on the pipeline and give content and label from the training data as input to train the model. After the model is trained it is possible to make predictions for both the training and validation data by using predict() with only content as input.

Naive Bayes (NB)

The Naive Bayes model has a simple and naive approach but can yield surprising accurate results (JavatPoint). It is based on Bayes theorem. We implemented a multinominal naive bayes classifier, which is suitable for text classification. We applied a tfidf transformer on the text/content in the articles for normalization of the words. For the actual implementation we used the sklearn packages naive_bayes, pipline and feature_extraction. Like with the SVM we called fit() on the training set and predict() afterwards.

Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is an advanced recurrent neural network (RNN) with an input gate, forget gate and output gate (Phi [2018]).

In our implementation we use TensorFlow Keras to tokenize and convert content to a sequence of wordindices and labels to a sequence of labels for both training and validation data. We define the layers with Sequencial() and input tf.keras.layers.Embedding, tf.keras.layers.Bidirectional(tf.keras.layers.LSTM) and tf.keras.layers.Dense. After this we configured the model for training by using compile() and trained the model with fit() with our train sequence, validate sequence, train label sequence and validate label sequence as well as number of epochs as input. Finally we were able to predict by predict() on both train and validation data with our train/validation sequences as input.

We found that five epochs was suitable for our model. We did not want to overfit our model by using too many epochs (GeeksforGeeks [2020]). On the other hand we wanted to train the data enough to obtain the best possible performance.

Quantifying the performance of the Fake News predictors

We began by splitting the original data set in a training and a test portion to avoid data leakage as mentioned in task 1.8. However, we still had to evaluate the model performance while developing, so we applied a 5-fold cross-validation strategy (80% train and 20% validation) similar to H. Ahmed et al. (2017 in Murari et al. [2021]. The size of the training data set consists of 291,945 articles while the validation data set is 72,985. The reason for doing cross validation is to evaluate our models performance across different training sets and ensuring good model generalization.

Model	Type	Precision	Recall	Accuracy	F1
1 Most common label (MCL)	train	NaN	0.0000	0.5410 (± 0.0005)	NaN
	validate	NaN	0.0000	0.5410 (± 0.0022)	NaN
2 Random	train	0.4588	0.4998	0.4999 (± 0.0008)	0.4784
	validate	0.4585	0.4988	0.4995 (± 0.0036)	0.4778
Random Forest (RF)	train	0.9481	0.8641	0.9162 (± 0.0011)	0.9043
	validate	0.9050	0.7864	0.8604 (± 0.0101)	0.8416
3 SVM	train	0.9733	0.6524	0.8322 (± 0.0010)	0.7812
	validate	0.9488	0.7223	0.8546 (± 0.0010)	0.8202
4 NB	train	0.9514	0.7146	0.8523 (± 0.0003)	0.8162
	validate	0.9455	0.6967	0.8424 (± 0.0011)	0.8022
5 LSTM	train	0.9808	0.9611	0.9735 (± 0.0018)	0.9708
	validate	0.9370	0.9161	0.9331 (± 0.0018)	0.9263

Table 2: The baseline and complex model's performance. The parentheses in the accuracy column indicates the std. dev.

Table 2 shows the different models' average performance of the five cross-validations with the standard deviation in parentheses. On a high level we see that the models are not over fitting to the training data. This is evident from the

figures as they are very stable between the training and validation set, which also indicates good generalization of the models. This is furthermore supported by the low standard deviations the models are having.

We have evaluated our models on the most widely used evaluation metrics: precision, recall, f1 and accuracy (Shu et al. [2017]). Our two worst performing models are baseline models Random, Most Common Label (MCL) while LSTM model is the best one achieving 93.31% accuracy in its predictions of the validation data set and a F1-Score of 92.63%. The models SVM and NB are also doing very well with accuracies of 85.2% and 84.2% respectively. This performance is inline with the Random Forest (RF) baseline model . Since the precision performance is similar across the RF, SVM, NB and LSTM models it is the better performance in recall that sets LSTM apart from the rest. In other words, LSTM is better at actually identifying when an article is fake news.

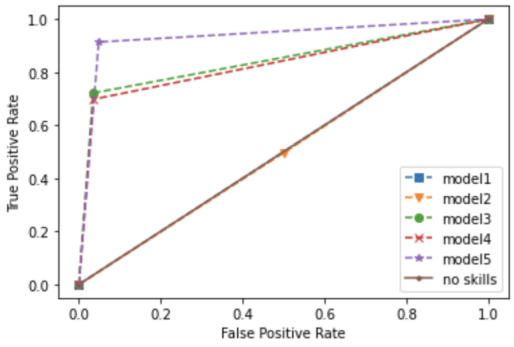


Figure 5: A ROC curve of the five models compared to zero skills.

Our LSTM model performance of 93.31% is very much in line with what other researchers have achieved (89% - 99.9%) (Murari et al. [2021]). These results is however based upon different data sets than what we have used.

4 Performance beyond the original dataset

The LIAR data-set is built quite differently from the "Fake News" data we trained our algorithm on. This is both in terms of columns and what 'content' conatins. While we have trained on the long-form FakeNewsCorpus articles, the LIAR data-set consists of very short statements, such as tweets or tv-ads.

The built-in classification of the LIAR statements consisted of *pants-fire*, *false*, *barely true*, *half-true*, *mostly-true*, and *true*. In order to have a balanced dataset with a more reasonable FAKE/REAL classification, we removed *barely true* and *half-true*, and labelled *pants-fire*, *false* as FAKE and *mostly-true*, *true* as REAL.

We used our best model (LSTM), which we had trained and tested on the Fake News Corpus (FNC), on the LIAR dataset along with a baseline of random guesses (see 2).
Unfortunately our algorithm didn't have an accuracy that was in any way better than random guesses, as seen in figure 3.

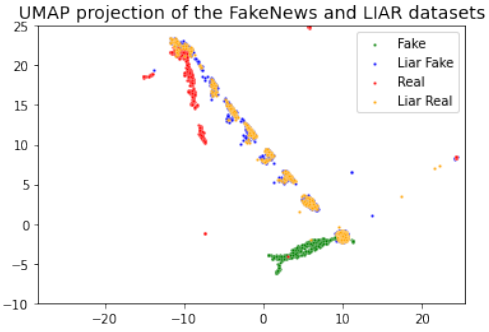


Figure 6: A visualization of part of the FNC data-set using supervised UMAP for dimensionality reduction. This is mainly to show that the FNC data should be relatively cleanly separable. It can also be seen that the LIAR data is not trivially separable in the same features.

We have a couple of different hypotheses as to why this has happened:

Models are overfitting In an attempt to adress this concern we tried training on less data with simpler models. In our testings we found no way to improve the performance on the LIAR-data set (but found a lot of ways to worsen it on the FakeNewsCorpus).

	LIAR	**FNC**
Predicted (LSTM Model)	0.507	0.933
Random baseline	0.491	0.500
Best baseline (RF)	0.448	0.871

Table 3: Accuracy of our model on the LIAR dataset and Fake News Corpus (FNC).

Models are not complex enough Increasing the number of trainable parameters on the LSTM (our most complex model by far), did not change anything significant except for the training time.

The data is simply too different Given that our accuracies are consisitently about 95% on both test and validation set for the Fake News Corpus, it seems likely that the problem our model is facing, is the difference between data from FNC and LIAR.

5 Discussion

1 Even though the performance on our test-set was quite good (as seen in section [3]), we unfortunately didn't see the same results when we tested it on the LIAR dataset (section [4]). It should be taken into consideration, that we had a relatively small subset of the FNC to work with [1], where articles labelled as Political, Junksci, and Conspiracy were included as well. Furthermore, the LIAR dataset had a very different categorization, with six different categories of truthfullness [Wang, 2017], which might have contributed to the discrepancy between the REAL/FAKE-labels of the two different data-sets.
As noted in section 4 the general construction of data is also significantly different in LIAR compared to the FNC. In table 4 we also see how articles in the FNC has a way higher mean of words compared to LIAR and generally a different distribution (see 8), which most likely will affect the performance of our algorithm.

	Mean	**Median**
FNC	513.6	329.0
LIAR	16.9	16.0

Table 4: Number of words per article/content.

2 Since datasets in general are widely different due to being created for very different purposes, it's not surprising to us that an algorithm trained on one type of text reacts very different to another type of text.

To improve performance of the Fake News predictor we created, it seems clear that texts of the same make-up should be taken into consideration during training, as we'd otherwise end up with an algorithm that can only be used on the particular type of text we trained it on.
Generally, if we want to predict fake news of a certain format, we will need to train our algorithm on news of that format.

As to whether further progress is gained mainly from better data or better models, we believe that data might reach a point, where we could hardly have better data (ie. no missing cells, all labels are correct, etc.), and therefore we assume that progress would mainly be gained from better models. However, we aren't quite there yet in terms of data, as "*Labeled data is not easily available that can be used for training the classifiers for detecting the fake news*" according to [Ahmed et al., 2021] as it's one of the main problems when creating algorithms to detect fake news. We believe this is mainly due to data collection not being very profitable in itself, therefore not necessarily highly prioritized for those who are not collecting data to create ML algorithms. This is noticeable in the Fake News Corpus which is labelled solely on domain without fact-checking articles, creating a large potential for mislabelled articles [Szpakowski, 2020]. Based on the existence of the universal approximation theorem, we assume that it is possible to find an approximated solution [Upadhyay, 2021], where an algorithm would be able to detect fake news perfectly.

Considering the fact that we steadily get better and better technology (processing power, memory and so on), there is no doubt that the model we spend several hours training, will be trained in minutes at some point in the future. This opens up a whole new world of thoroughly trained models. At the same time, this might make overfitting an even more common problem than it already is, which would require students and researchers outside this field to be even more aware of it, just like p-values.

References

GoogleTrends. Search results for wikinews. URL https://trends.google.com/trends/explore?date=all&q=%2Fm%2F04dcty.

George V Jose. Predicting sequential data using lstm: An introduction, 2019. URL https://towardsdatascience.com/time-series-forecasting-with-recurrent-neural-networks-74674e289816.

GeeksForGeeks. Support vector machine in machine learning, 2020. URL https://www.geeksforgeeks.org/support-vector-machine-in-machine-learning/.

JavatPoint. Naïve bayes classifier algorithm. URL https://www.javatpoint.com/machine-learning-naive-bayes-classifier.

Desmond Elliott. Learning from data 2, supervised classification. URL https://absalon.ku.dk/courses/56568/files/5925296?module_item_id=1636649.

Michael Phi. Illustrated guide to lstm's and gru's: A step by step explanation. 2018. URL https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21.

GeeksforGeeks. Choose optimal number of epochs to train a neural network in keras. 2020. URL https://www.geeksforgeeks.org/choose-optimal-number-of-epochs-to-train-a-neural-network-in-keras/.

Choudhary Murari, Jha Shashank, Saxena Deepika, and Singh Ashutosh. A review of fake news detection methods using machine learning. 2021. URL https://www.researchgate.net/publication/351775335_A_Review_of_Fake_News_Detection_Methods_using_Machine_Learning.

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake news detection on social media: A data mining perspective. *SIGKDD Explor. Newsl.*, 19(1):22–36, sep 2017. ISSN 1931-0145. doi:10.1145/3137597.3137600. URL https://doi.org/10.1145/3137597.3137600%7D.

William Yang Wang. "liar, liar pants on fire": A new benchmark dataset for fake news detection, 2017. URL https://aclanthology.org/P17-2067.pdf.

Alim Al Ayub Ahmed, Ayman Aljabouh, Praveen Kumar Donepudi, and Myung Suh Choi. Detecting fake news using machine learning : A systematic literature review. 2021. URL https://arxiv.org/abs/2102.04458.

Maciej Szpakowski. Fake news corpus readme, 2020. URL https://github.com/several27/FakeNewsCorpus/blob/master/README.md#limitations.

Khushee Upadhyay. Beginner's guide to universal approximation theorem, 2021. URL https://www.analyticsvidhya.com/blog/2021/06/beginners-guide-to-universal-approximation-theorem/.

6 Appendix

Know your data

'**Million**' has the columns:

Index, *bigint*	Unnamed:0, *text*	Id, *text*	Domain, *text*
Type, *text*	Url, *text*	Content, *text*	Scraped_at, *text*
Inserted_at, *text*	Updated_at, *text*	Title, *text*	Authors, *text*
Keywords, *double precision*	Meta_keywords, *text*	Tags, *text*	Summary, *double precision*
Source, *double precision*			

The four dimension tables '**dim_authors**', '**dim_domain**', '**dim_type**', and '**dim_meta_keywords**' with their respective columns:

Dim_authors	Dim_domain	Dim_type	Dim_meta_keywords
authors, *text*	domain, *text*	type, *text*	meta_keywords, *text*
authors_id, *int*	domain_id, *int*	type_id, *int*	meta_keywords_id, *int*

'**fact_fake_news**' has the columns:

Id, *text*	Domain_id, *int*	Type_id, *int*	Author_id, *text*
Url, *text*	Content, *text*	Scraped_at, *text*	Title, *text*
Keywords, *double precision*	meta_keywords_id, *int*	Meta_description, *text*	Tags, *text*
Summary, *double precision*			

Error on line 3586125

```
datascience=# COPY raw_data(
datascience(# id, kid, domain, type, url
datascience(# ,content, scraped_at, inserted_at, updated_at
datascience(# ,title, authors, keywords, meta_keywords
datascience(# ,meta_description, tags, summary, source
datascience(# )
datascience-# FROM 'C:\Users\Martin\Workspace\ds\2022-04-18-medium.csv'
datascience-#
datascience-# WITH (format csv, header, DELIMITER ',',ENCODING 'UTF8');
ERROR:  missing data for column "tags"
CONTEXT:  COPY raw_data, line 3586125: "6453,131316,ecowatch.com,political,https://www.ecowatch.com/confirmed-american-a
cademy-of-pediatrics..."
datascience=#
```

Figure 7: Error on line 3586125

Plots from fig 4a

```
SELECT COUNT(Title), SUBSTRING(Date, -6, 4) F
ROM wikinews
GROUP BY SUBSTRING(Date, -6, 4)
ORDER BY SUBSTRING(Date, -6, 4) asc
```

Plots from fig 4b

```
SELECT count(*), length(Sources) -
length(replace(Sources, 'xa0', 'xa')) + 1
FROM wikinews GROUP BY length(Sources) -
length(replace(Sources, 'xa0', 'xa'))
```

Establish a baseline

Implementation of meta-keywords

There were some discussion as for the optimal way to take the meta-keywords into consideration. We ended up simply adding the words in both the start and the end of the article to give them additional "presence".

Performance beyond the original dataset

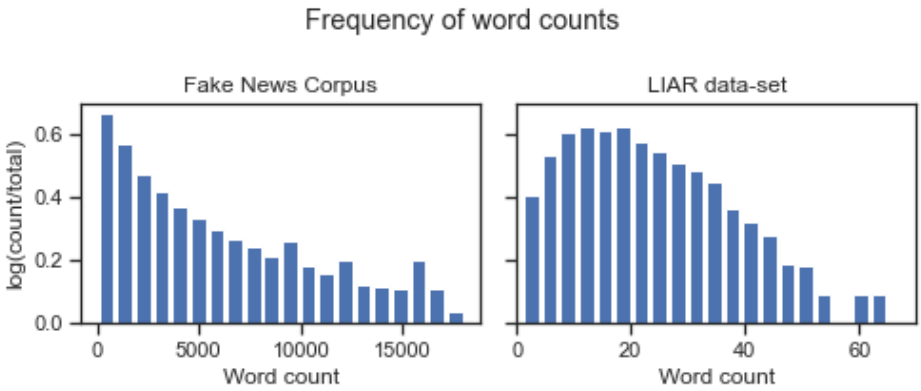


Figure 8: Distribution of words pr. article/content from the Fake News Corpus and the LIAR dataset.