DATA SCIENCE FINAL PROJECT

GROUP 12

Xiqing Mao
Department of Biology
University of Copenhagen
tls868@alumni.ku.dk

Zelin Li
Department of Biology
University of Copenhagen
qzj475@alumni.ku.dk

Lu Wang
Department of Computer Science
University of Copenhagen
vqn532@alumni.ku.dk

July 18, 2021

Contents

1 Know your data

1.1 FakeNewsCorpus

1. Schema setting is crucial for database designing. Fig. 1 is the ER-diagram of FakeNewsCorpus dataset, rectangle stands for entity, ellipse stands for attribute while diamond for relation; the primary key marked in red, along with green for foreign key, followed abbreviations used in the diagram: `art` = `articles`, `aut` = `authors`, `w_by` = `written_by`, `art_id` = `id`, `meta_k` = `meta_keywords`, `meta_d` = `meta_description`, `sc_at` = `scarped_at`, `ins_at` = `inserted_at` and `upd_at` = `updated_at`.

Logically stored data will provide a great query experience, and database normalization will minimize data anomalies and redundancies. The suitable functional dependency enables each relation in our database to satisfy BCNF. Hence, there is no violation of data normalization in our database.

For implementation, `PostgreSQL` was utilized. Instead of loading the whole Fake News Corpus into the database, a randomly selected subset with 1000000 articles was loaded.

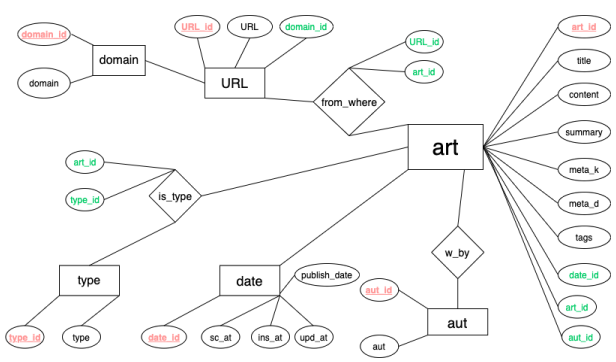

Figure 1: ER-diagram of FakeNewsCorpus

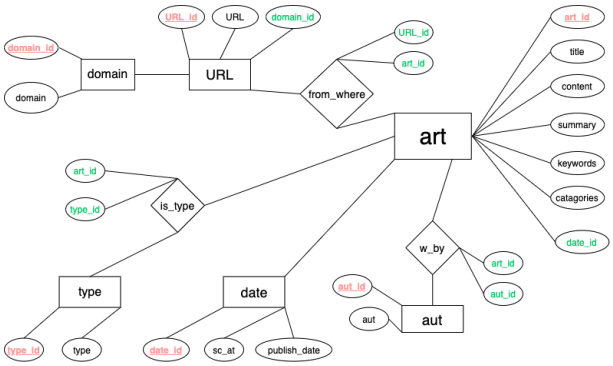

Figure 2: ER-diagram of Wikinews

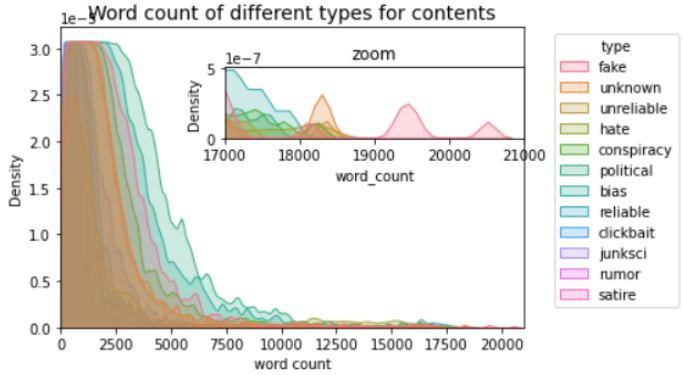

Figure 3: FakeNewsCorpus articles word counts

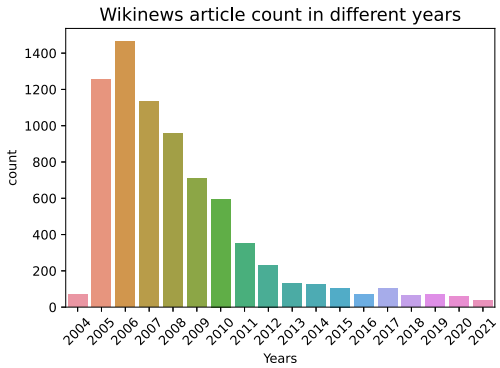

Figure 4: Wikinews article counts

2. Inherent problems in the dataset:

1) A portion of the attributes are empty (N/A or NULL), including `tags`, `meta_description`, `meta_keywords` and even `type` (38120 articles without it). Attributes `keywords`, `summary`, `source` are NULL for the entire dataset. We only removed `source` and `keywords` from the input; `summary` column is remained, which we assume `keywords` can be replaced by `meta_k`, and `source` can be viewed as a property of domain or URL;

2) Data types are not formatting consistently in attribute `id`, values are not integers but floats or characters. 999934 articles left after removing;

3) In column `content`, the containing are partially duplicates. For some data, all the attributes even entirely overlapped, only distinguish with a different `id`. We removed 95023 duplicates, with 904911 articles remaining;

4) We consider time details are not meaningful for further analysis, therefore we use DATE rather than TIMESTAMP, which lead to identical values for `sc_at`,`ins_at`,`upd_at`.

3. Some dataset observations:

1) Most articles have a length below 5000 words, in contrast, fake news is 18000 above. The results are illustrated in Fig. 3;

2) Articles in a same `domain` have the same `type`.

1.2 Wikinews

4. `BeautifulSoup` and `Regular Expression(re)` are used to locate 'next page' URL, then iterate on all pages on Wikinews, at last all articles' URL are fetched into a URL set. By iterating on these sets, all articles could be scraped. For each article, its `URL`, `title`, `content`, `publish_date` and `categories` were scraped through `re`, while its `summary` and `keywords` were simultaneously generated through a Python package `newspaper`, which is the package that were used to generate FakeNewsCorpus (it embedded NLP). The overall scraping process was fluent and successful, this may due to no anti-crawling techniques were used in Wikinews.

5. We use `newspaper` method to generate metadata like `keywords`, `summary`, `categories` and `publish_date` as well, which are hard to scrap directly from Wikinews. All `aut` were marked as 'wikinews', all `domain` were 'wikinews.org', and all `type` were 'political'. The schema of Wikinews dataset is almost the same as FakeNewsCorpus dataset (its ER-diagram is shown on Fig. 2), but with some changes where were shown on Fig. 2.

6. We have got some basic statistics on Wikinews, the total number of articles is 7576. Fig. 4 shows the number of articles published each year, Showing that most of articles in Wikinews were published from 2005-2012. The SQL for generating it is:

```
SELECT date_trunc('year', publish_date) AS year_group, COUNT(art_id) AS annually_count
FROM wikinews.art INNER JOIN wikinews.date USING(date_id) GROUP BY year_group
```

1.3 Integration

7. A schema for SQL VIEW to integrate two sources was created: Attributes with same name among two sources were merged, except `domain` and `aut`; `keywords` were mapped to `meta_k` since they are both multiple words that resembles keywords; `categories` were mapped to `tags` since they both contain words that resemble the tag; all other properties that can not find any common property in another source were excluded (`ins_at`,`upd_at` and `meta_d`).

8. Content and title from both sources have remained for the remaining part of the project, the reason of others were not included are: `domain` is type-specific for both sources, which will certainly introduce bias; `sc_at` do not refer to the published date of an article, so its information may be misleading; `aut`,`summary`,`keywords`,`categories` are not present in many articles, including them may introduce bias as well.

9. Based on the description of types on FakeNewsCorpus github, in our dataset, `fake, bias, conspiracy, junksci, satire, hate` and `rumor` were classified as fake news, `political`,`clickbait` and `reliable` were classified as real news. `clickbait` was treated as true news because only its headlines, descriptions and images are unreliable, while the contents consider as credible to some degree. `unknown` was deleted from dataset since the type is unclear. `unreliable` was deleted from dataset since its contents require further verification.

2 Logistic regression as a baseline

2.1 The baseline

The `content` were not cleaned in baseline.

Logistic regression was chosen as the baseline because it:

1. Simple to implement and widely used as a baseline in the industry;
2. The output is a probability, which means a series of continuous numbers. This can not only predict fake news but also score the news.
3. Good explainability.
4. Low computational cost and fast speed. Logistic regression can handle the classification problem with a linear relationship between the feature and the label, and the accuracy is high. The other three models (Nearest Neighbors, Random Forest, Neural Network) were also compared with Logistic regression. Table 1

shows the advantages of logistic regression. SVM was also used to make predictions, but in the end, it was abandoned because of slowness.

The dataset is `1mio-raw.csv` and news collected from the "Politics and Conflict" section of the Wikinews.

It was sampled 10% and was used an 80/10/10 training/val/test split with random seed 42. TF-IDF was used to vectorize articles, with `max_features = 256`.

Parameters: `penalty='l1',solver='saga', max_iter=10000`.

2.2 Baseline with metadata

Among those metadata, only `title` was believed makes sense. The reasons why other features are meaningless are as follows.

1. `domain`: in FakeNewsCorpus, the relationship of `type` between `domain` is one-to-one.
2. `url`: each news' URL is unique.
3. The time of `scrap, insert, update`: unrelated with types.
4. Other fields, like `author, summary, description`: insufficient data.

Table 1 shows the accuracy of with and without the label, which confirmed our point of view. However, the performance improvement of including the title is very limited. This could be caused by: the words of the title are general and universal, the title is usually short, and the title of `Clickbait` news looks like from fake news, but the content is generally credible.

Table 1: Accuracy of different models

	Nearest Neighbors	Random Forest	Neural Network	Logistic regression
With Title)	0.7678	0.6749	0.7541	0.8156
Without title	0.7623	0.6785	0.7311	0.8052

3 Create a Fake News predictor

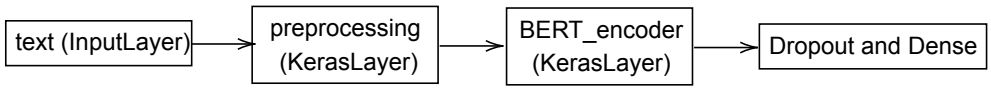

Figure 5: Architecture of model

Fig. 5 shows the architecture of predictor, which uses BERT model(bert/bert_en_uncased_L-12_H-256_A-4). Emails, URLs, any number, spaces, special characters and stop words were removed from articles and they were stemmed before feeding to the input layer. The prepossessing layer transform text to format expected by the BERT model: `input_word_ids`, `input_mask` and `input_type_ids`, which refer to the word embedding ids, the indicator of content and the padding, and the indicator of token sentences.

We followed suggestions of hyperparameters from original BERT paper:

1. Batch size: 16, 32
2. Learning rate (Adam): 5e-5, 3e-5, 2e-5
3. Number of epochs: 3, 4

The effect on the performance of batch size is quite small, so 32 was chosen for faster training. After trying other parameter combinations, learning rate = 2e-5, epochs = 2 and seq_length = 256 show the best performance. The history of performance did not illustrate since it was only trained 2 epochs.

A small dropout(0.1) was used to prevent overfitting, due to BERT is a deep model.

3.1 Predictor against baseline

With the same dataset, the test performance of the predictor is 0.9123, which is an increase of 13.3% compared to the test performance of baseline (0.8052).

Table 2: Accuracy test

	Precision		Recall		F1-score		Support
	Predictor	Baseline	Predictor	Baseline	Predictor	Baseline	Both
Fake	0.93	0.84	0.93	0.87	0.93	0.85	5387
Real	0.88	0.75	0.88	0.69	0.88	0.72	2948
Accuracy					0.91	0.81	8335
Macro avg	0.90	0.79	0.90	0.78	0.90	0.79	8335
Weighted avg	0.91	0.81	0.91	0.81	0.91	0.81	8335

3.2 Why BERT

BERT, a popular language representation model in recent years, is used to finish this task.There is some improved version of BERT, such as RoBERTa and XLNet. We tried RoBERTa with the same dataset on 8 CPUs and 1 GPU, but it cost around 23 hours for 6 epochs. We also tried other methods, the results of the testset of BERT is higher than them (details in Tab. 3), after weighing speed, size, and quality, we decided to use BERT (BERT-Large).

Table 3: Different models best performance we have tried

	LSTM	fasttext	RoBERTa-Base	RoBERTa-Large*	BERT-Base	BERT-Large
Accuracy	0.8851	0.8518	0.9057	0.7757	0.9081	0.9123

* uncompleted, only run 517/2083 of 1 epoch

BERT improves many disadvantages of potential alternatives. The TF-IDF algorithm used in the baseline model does not consider the influence on differences of text by location of keywords, and the IDF of some rare words will be relatively high. In addition, the output (sparse matrix) brings higher computation costs. Some Word Embedding techniques such as word2vec and FastText turn the sparse matrix into a dense matrix. But they cannot handle polysemous words. Another model ELMO provides a simple and elegant solution for it. ELMO can learn Word Embedding of a word in advance by using the language model, which means the Word Embedding representation of a word can be adjusted according to the semantics of the context word. However, the LSTM used by ELMO is weak in feature extraction capabilities.

BERT uses an advanced version of attention, namely transformer, as the feature extractor to solve the problem, which is the earliest input sequence cannot be decoded by LSTM well when facing long sequences. The idea of Attention is to select the feature that currently has the greatest impact on each decoding moment. The three vectors `Q, K`, and `V` required by the self-attention used by Transformer are obtained from the three linear transformations of the input, and each `Q` can be multiplied by the original `K` to obtain the corresponding `V`. Therefore, it is possible to parallelize and obtain the semantic information of a long sequence, avoiding the loss of information transmitted over long distances due to forward propagation in RNN/LSTM.

BERT also uses a bidirectional language model, which means it is better than the GPT model that only uses the context-before data. In addition, BERT uses more massive data than GPT for self-supervised training. The good performances of BERT in various competitions prove that a very deep model can significantly improve the accuracy of the NLP task, and this model can be pre-trained from the unlabeled dataset.

The pre-trained + fine-tuning design allows people to easily modify the input and output ports of the BERT, enabling adaptation to a wide variety of downstream NLP tasks. For this news classification task, we only need to add start and end symbols to the input, then connect an activation function to the output.

In conclusion, BERT is a milestone model of NLP. It draws on the advantages of its predecessors and improves their shortcomings, so it was chosen to predict fake news.

4 Performance beyond the original data-set

Table 4 shows the accuracy of baseline and our predictor on different datasets. The results of baseline and predictor on the original dataset and LIAR dataset are similar. Most of the predictions of the two models are correct on the

original database, but both their performance on the LIAR database is poor, with around 50% error rate. For the Kaggle database, the accuracy of the predictor is nearly 52.99% higher than the baseline.

Table 4: Performance on different datasets

	Original dataset	LIAR dataset *	Kaggle leaderboard
Baseline	0.8052	0.4923	0.5207
Fake News predictor	0.9123	0.4792	0.7699

* We use `test.tsv` with only `FALSE` and `TRUE` included

5 Discussion

5.1 Discrepancy

Models' accuracy on LIAR is extremely low, which might due to its contents contain so few words; from Fig. 6 we can see the frequency of articles word length in different datasets. So LIAR's performance is worse than the other two.

Since our predictor use seq_length=256, the original and Kaggle dataset after cleaning still can have enough (256) tokens for most cases. But LIAR's content average words is far less than 256 so that after cleaning, the performance of our predictor even worse than baseline (without cleaning). For example, after cleaning, some LIAR content only remained 3 words.

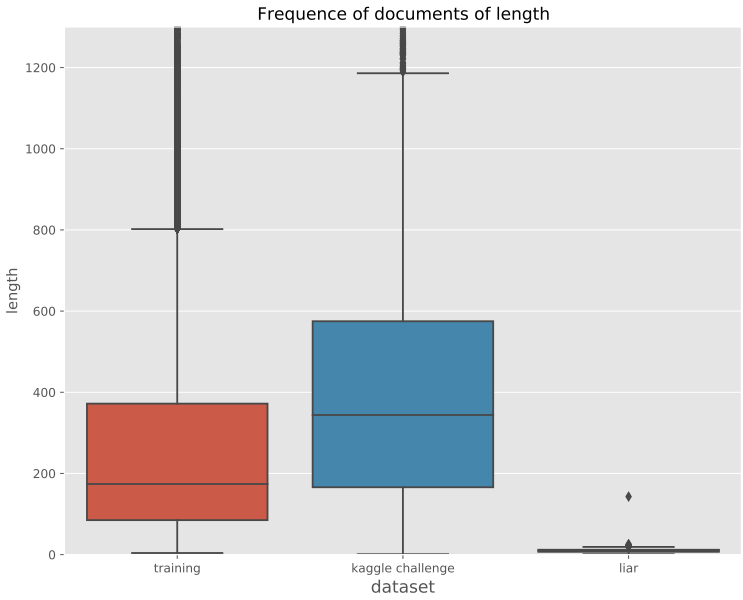

Figure 6: Word length statistics of different dataset

As for the Kaggle dataset:

1) Its label technique may differ from the training. Bias can be introduced to both datasets due to labeling, for example, in the original dataset, `type` is determined by its `domain`, but it is doubtful to do so since fake news preference words could occur in those 'reliable' domain articles. And probably the Kaggle dataset does not classify `type` in this way, even it can opposite (a domain Kaggle considers as real while ours as fake).

2) From Fig. 4 we know that data sources have their unique years distribution, so the training data years distribution may be different from Kaggle, so the words preference of fake news of them are different too.

5.2 Bias and variance

The discrepancy between performance on different datasets aren't surprise since the data form and classification techniques are different, the labels can not uniform, for example, 'reliable' doesn't mean 'real' or 'true'. We did surprise by Kaggle's result, which is under 80%.

Our model is with high bias with low variance since we have selection bias which subjective selected the initial dataset to train with, that we build the model with intention bias to predict from limit data.

To improve the performance, we can try to fine-tune the model, change the model, add metadata, or simply using a larger training set. For example, we can change model 'BERT-Large' to 'Roberta-Large' with bigger training sets that integrate metadata `type` and `author`.

Progress will be driven primarily by data. More accurate type definition and reliable labeling on type are the most important things. If the labels are uniform and unambiguous, then a relatively good predictor can produce by almost any model.

This is almost not a solvable problem, because it is hard to distinguish the fake news even for humans. Bias could be introduced through the subjectivity from label makers, but it can be improved for more objective labeling through censor on labeling or multiple people mark labels.

5.3 Furthermore

We follow the software development norm to process our project, we use GitHub to store our project and make branching for editing with diverse changes ready for later merging. But we found that it is hard to cooperate based on Jupyter since it can't pull-request solved divergence automatically. We found it's hard to transfer data on the different platforms too (we used Colab, PC, and computer clusters).

We use Black-box-testing to find the metrics of the model's precision, F1-score, and accuracy. For instance, to pick a baseline model, we run training datasets on different models, then select the one with the highest accuracy among the variants to establish our baseline. White-box-testing can not be applied directly to our models, since they need to be trained by data.

6 Code

`https://github.com/Xiiqiing/DS_project`