|  |
| --- |
| **Fake (Sarcasm) News Text Classification** |
|  |
| **732A81 Text Mining Project** |
|  |
|  |
|  |
|  |

Abstract

Fake news contains incorrect information about a certain entity that could mislead people in taking erroneous decisions or which may cause unrest in some countries. The globalization and digitalization of the world has only worsened the situation by introducing innumerable news sources. Since, the veracity of everyday news plays a central part in people’s lives, we aim to research about different machine learning models and make an analysis of the research related to fake news detection. We have used transformer-based models which is current state-of-the-art for many machine learning problems, especially, for natural language processing. This is a binary text classification problem, hence we have multiple metrics such as accuracy, precision, recall, F1-score and ROCAUC score to evaluate our model.

Introduction

Manual fact checking can solve some of the problems relating to fake news based on technical background check and contacting authorities but is too slow to cover massive information spreaders such as social media platforms [4]. It’s the sheer range of fake news spread over different communication networks that has posed a major challenge for us. In this paper, we have aimed to leverage artificial intelligence tools based on natural language processing and text mining to classify news based on its satirical property. Natural language processing has been a topic of interest for a very long time and with more compute power, we have created models that are able to match human level accuracy. We have analyzed different models that could be trained on text data and be tested on unseen data. The purpose of this project is to create a solution that could be used to detect and filter out news headlines containing fake news [5].

Theory

Supervised Learning

The supervised learning method utilizes the training data to understand and learn the information about the intended job of the model in order to foresee the hidden activities and patterns in the data [6]. The amount of data depends on the method we use to train our model such as transfer learning or training the model from scratch. After training, we evaluate our model on an unseen data normally called testing data. It is imperative that we do not let the model get the knowledge about test data as it would obviously lead to bias, and our model could produce erroneous results.

Natural Language Processing

Natural Language Processing or NLP for short, is a broad field that includes methods from classifying text corpus to a predefined classes to understanding a dialogues between a human and a chatbot. The study of natural language processing has been around for almost 5 decades and continuous improvements in the field has facilitated understanding of linguistics with the rise of computers [7]. It is extremely challenging to work with NLP because computers do not understand text or speech like humans do, and this requires research in encapsulating the nuances of language (text or speech) in mathematical functions and models. The benefits of NLP for businesses outweighs its difficulty in implementing such solution as it is so critical to businesses to tap on to large volumes of not just numerical data but text data like social media, customer support tickets, online comments, online reviews and more [8].

Deep Learning Text Classification

Text classification is the principal test in natural language processing. It is the procedure of designating pre-defined labels for text and has diverse applications such as sentiment analysis, topic labeling, dialog act classification, question answering and much more [9]. In this project we are leveraging the help of deep learning algorithms instead of classical machine learning algorithms even though deep learning is a subset of machine learning, we have different processes and techniques to implement deep learning algorithms. These algorithms are proven to be more robust and produce better results when treated with huge amounts of data. They have the capability of understanding the features and patterns of the text data automatically in contrast with classical machine learning algorithms where we had to create meaningful features from the text data.

Diagram, text

Description automatically generated

Figure 1: Flowchart describing lifecycle of text classification with classical methods in each module. It is important to extract essential features for standard machine learning methods, but they can be extracted automatically by deep learning algorithms.

Transfer learning

Since the inception of well-defined machine learning models, it has opened an avenue for variety of industrial applications each with its own data requirements. Transfer learning is a method where a machine learning model is reused for a downstream task (similar task but with some constraints). Many smaller tasks can get very high accuracy by using a pre-trained model on similar data and modifying the architecture of the model for the small task [10]. Basically, pretrained models use features extracted from previous training on any kind of data they have been trained for but mostly that is industrial level data (size: petabytes of data) and use that knowledge to understand patterns of off different and unique datasets, which in our case, are the news headlines (text data). This saves vast compute and time resources required to develop neural network models on these problems. We have only used supervised pre-trained deep neural networks trained primarily on BooksCorpus (800 M words) and English Wikipedia (2500 M words) and then we fine-tuned the model for our dataset.

Chart

Description automatically generated with medium confidence

Figure 2: Traditional Learning vs Transfer Learning

Data

Data Description

The data is called ‘News Headlines Dataset for Sarcasm Detection’ on Kaggle (Data science and machine learning platform) [11]. It is primarily used for the task of sarcasm and fake news detection. Past studies in Sarcasm detection mostly makes use of Twitter datasets collected using hashtag-based supervision but datasets are noisy in terms of labels and language. Furthermore, many tweets are replies to other tweets and detecting sarcasm in these requires the availability of contextual tweets. To overcome the limitations related to noise in Twitter datasets, this News Headlines dataset for Sarcasm Detection is collected from two news website. TheOnion aims at producing sarcastic versions of current events and we collected all the headlines from News in Brief and News in Photos categories (which are sarcastic). We collect real (and non-sarcastic) news headlines from HuffPost.

* 1. Data Format

The data is primarily in .json (Java script object notation) format that has python dictionary-based structure defining the elements in each sample. The data could be read using following method in python:

Text

Description automatically generated

Figure 2: Above code snippet used to read the data in python but also can be read using more advanced pandas library and converted into pandas Data Frame.

* 1. Data Content

Each record consists of three attributes:

* **headline**: the headline of the news article
* **is\_sarcastic**: 1 if the record is sarcastic otherwise 0
* **article\_link**: link to the original news article. Useful for collecting supplementary data.

An example sample looks something like below in .json format. It is similar to a python dictionary and can be accessed with the help of keys.

* “root”: {
* "article\_link": "https://www.huffingtonpost.com/entry/versace-black-code\_us\_5861fbefe4b0de3a08f600d5",
* "headline": "former versace store clerk sues over secret 'black code' for minority shoppers",
* "is\_sarcastic": 0

}

Data Preprocessing

Text data is notorious for containing a lot of noise because of different linguistic styles, slangs, special characters, nuances and much more. One of the most important step in any NLP task is to preprocess the data and format it according to the model input requirements. In our case, we have also done several preprocessing steps before training our algorithm.

* **Stopwords Removal**: A stop word is a frequently occurring word in a text that has minor or no effect on the meaning of the sentence. It does not contribute to the context of the text and hence search engines have been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query [12]. NLTK is one of the libraries that has garnered much attention while preprocessing text data as it stores a list of stopwords stored in 16 different languages and we use the same in our use case. We have also used regex (regular expressions) library to remove any special characters and punctuation from the text.
* **Lemmatization**: For grammatical reasons, documents/texts may use different forms of a word, such as play, plays, played or playing. Additionally, there are families of derivationally related words with similar meanings, such as democracy, democratic, and democratization. In many scenarios, it appears as if it would be useful for a query for one of these words to return documents/texts that contain another word in the set [13]. Lemmatization usually refers to the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma.

Graphical user interface, text

Description automatically generated

Figure 3: Example of lemmatization on a sentence.

In our case, we again use NLTK ‘wordnet’ lemmatizer to preprocess our data. It is a large, freely, and publicly available lexical database for English language aiming to establish structured semantic relationships between words.

* **Data Splitting:** We are performing supervised learning and for that we must split our data into train/valid/test. The ratio of the split has been set to 80:10:10 which means 80% of the whole dataset is train, 10% of the left is validation set and the remaining is testing set. We train our model on training set, tune model’s hyperparameters which evaluating on validation set and finally test our model on testing set. We have total 55328 rows (samples) and 3 columns (features) in our original dataset. It looks something like below after the data split:

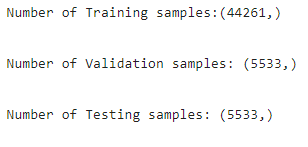


Figure 4: Data split between train/valid/test set.

Method

We approached the problem using a standardized structure similar to any Natural language processing task where after data preprocessing we build our models and set the hyperparameters. For this problem, we are going to take advantage of certain libraries, tools, and transformer-based models.

Transformers

We will be using huggingface [14] implementation of pre-trained transformers model in our project. The transformer in NLP is an ingenuous approach that aims to solve sequence-to-sequence tasks while handling long-range dependencies with ease. One of the major contribution in these models is the concept called self-attention mechanism which helps the model create similar connections but within the same sentence, it allows us to focus on parts of out input sequence while we predict our output sequence.

Diagram

Description automatically generated

Figure 5: Transformer architecture

It contains one encoder block which maps an input sequence of symbol representations to a sequence of representations. And decoder blocks which given z, generates an output sequence of symbols one element at a time.

BERT Tokenizer

BERT stands for Bidirectional Encoder Representations from Transformers and is a language representation model by Google [15]. It is one of the most popular and go to transformer-based architecture for Natural language processing tasks. Simply put, it is a stack of bunch of Transformer’s Encoder. We shall be using different implementations of BERT for experimentation purposes and produce results containing different metrics as discussed earlier. It oversees preparation of text inputs for a model into appropriate numerical data which the model can understand and train on it. Basically, it converts text into a sequence of tokens, create a numerical representation of the tokens, and assemble them into tensors.

Graphical user interface, application

Description automatically generated

Text, application, chat or text message

Description automatically generated

Figure 6: The tokenizer using huggignface tansformer library and calling Autotokenizer.frompretrained method for ‘bert-base-cased’ model.

The tokenizer returns a dictionary with three important items:

* [input\_ids](https://huggingface.co/docs/transformers/glossary#input-ids): are the indices corresponding to each token in the sentence.
* [attention\_mask](https://huggingface.co/docs/transformers/glossary#attention-mask): indicates whether a token should be attended to or not.
* [token\_type\_ids](https://huggingface.co/docs/transformers/glossary#token-type-ids): identifies which sequence a token belongs to when there is more than one sequence.

These elements are finally converted to actual tensors that get fed to the model.

Fine-tuning BERT on our custom dataset

Fine-tuning is defined as utilizing the model architecture, pre-trained on a bigger dataset, to do various machine learning tasks such as classification, segmentation, generation by swapping out the appropriate inputs or outputs. To train-task specific models, we add an extra output layer to existing BERT and fine-tune the resultant model. Essentially, we do not change the central part of the BERT model but tweak input/output layers which allows change in only minimal number of parameters which need to learned from scratch making the procedure fast, cost and resource efficient.

Diagram

Description automatically generated

Figure 7: Single sentence classification task using BERT

We shall be using 3 different BERT implementations for our downstream task:

* BERT base (uncased) – Pretrained model on English language using a masked language modeling (MLM) objective [16]. This model is uncased: it does not make a difference between English and English.
* DistilBERT base uncased – This model is a distilled version of the BERT base model [17]. It is a transformer model, smaller and faster than BERT, which was pretrained on the same corpus in a self-supervised fashion, using the BERT base model as a teacher.
* ALBERT: A Lite BERT for Self-supervised Learning of Language Representations – It is a model that uses absolute position embeddings [18]. It presents two parameter-reduction techniques to lower memory consumption and increase the training speed of BERT: Splitting the embedding matrix into two smaller matrices and Using repeating layers split among groups.
* DeBERTa: Decoding-enhanced BERT with Disentangled Attention - It builds on RoBERTa with disentangled attention and enhanced mask decoder training with half of the data used in RoBERTa [19]. This is the DeBERTa large model fine-tuned with MNLI task.

**Cross-referencing:** To add a cross reference to a figure or table:

* Place the mouse pointer at the location where you wish to add the cross-reference.
* Click on the **Insert** menu, (then click **Reference**), and then **Cross-reference** in the **Links** panel.
* In the **Cross-reference** dialog box, click the caption to which you are building the text reference.
* For a figure, under **Reference Type**, click **Figure**.
* Under Insert Reference To, click Only Label and Number, then click OK.
* As much as possible, fonts in figures should conform to the document fonts (this is not the case in the example figure).

This is an example reference to Figure 1.

Hyperlinks

Within-document and external hyperlinks are indicated with Dark Blue text, Color Hex #000099.

References

To create hyperlinks between citations and references, as you insert each full reference in the References section, highlight it and then select Insert, Bookmark. Link back to the reference from its citations in the text by highlight the citation, right clicking, and selecting Insert, Cross-Reference, then selecting the Bookmark you’ve saved. Highlight the citation again to give make it dark blue (included in this theme), if it is not automatically applied. If there are problems saving the hyperlinks when you convert the document to PDF, use an online converter such as <http://go4convert.com>.

Citations

Citations can be created by creating in-document hyperlinks to bookmarks you’ve created. Go to Insert / Hyperlink / This Document / Bookmarks, and select your bookmark.

* 1. Equations

An example equation is shown below:

(1)

To add new equations, authors are encouraged to copy this existing equation line, and then replace with the new equation. The numbering and alignment of equation line elements is automatic. To update equation numbering, press **Ctrl-A + F9**. Note: this will only update the number to the right of the equation; to update numbering within the text you must create a cross-reference.

**Cross-referencing:** To create a cross-reference for an equation:

* Create a bookmark for it.
* Select the number to the right of the equation. Go to **Insert**, **Bookmark** (in the **Links** panel),andthen create a name for your equation. Press **Add** to create the bookmark.
* To refer back, place the mouse pointer at the location where you wish to add the cross reference.
* Go to **Insert, Cross-reference** (in the **Links** panel).In the dialogue box, select **Bookmark** and **Bookmark Text** from each dropdown list. Uncheck **Insert as Hyperlink**, then click **OK**.
* This will make it such that whenever a new equation is added, the references to the equation will be updated when **Ctrl-A + F9** is pressed.
* This an example cross-reference to Equation 1.

Appendices

Appendices, if any, directly follow the text and the

references. Letter them in sequence and provide an informative title: **Appendix A. Title of Appendix**.

1. MS Word STREAM Tools

This Microsoft Word file was updated in 2016 with STREAM Tools, designed for creating well-formatted reports and papers with Microsoft Word (Mamishev, 2010; Mamishev, 2013).

Limitations

ACL 2023 requires all submissions to have a section titled “Limitations”, for discussing the limitations of the paper as a complement to the discussion of strengths in the main text. This section should occur after the conclusion, but before the references. It will not count towards the page limit. The discussion of limitations is mandatory. Papers without a limitation section will be desk-rejected without review.

While we are open to different types of limitations, just mentioning that a set of results have been shown for English only probably does not reflect what we expect. Mentioning that the method works mostly for languages with limited morphology, like English, is a much better alternative. In addition, limitations such as low scalability to long text, the requirement of large GPU resources, or other things that inspire crucial further investigation are welcome.

Ethics Statement

Scientific work published at ACL 2023 must comply with the ACL Ethics Policy.[[1]](#footnote-1) We encourage all authors to include an explicit ethics statement on the broader impact of the work, or other ethical considerations after the conclusion but before the references. The ethics statement will not count toward the page limit (8 pages for long, 4 pages for short papers).

Acknowledgments

This document has been adapted by Jordan Boyd-Graber, Naoaki Okazaki, Anna Rogers from the template for earlier ACL, EMNLP and NAACL proceedings, including those for EACL 2023 by Isabelle Augenstein and Andreas Vlachos and EMNLP 2022 by Yue Zhang, Ryan Cotterell and Lea Frermann.

References

1. <https://medium.com/mlearning-ai/fine-tuning-bert-for-tweets-classification-ft-hugging-face-8afebadd5dbf>
2. <https://www.thepythoncode.com/article/finetuning-bert-using-huggingface-transformers-python>
3. Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).
4. <https://edps.europa.eu/press-publications/publications/techsonar/fake-news-detection_en>
5. Monther Aldwairi, Ali Alwahedi, *Detecting Fake News in Social Media Networks, Procedia Computer Science,*

Volume 141, 2018, Pages 215-222, <https://doi.org/10.1016/j.procs.2018.10.171>.

1. <https://iopscience.iop.org/article/10.1088/1757-899X/1099/1/012040/pdf>
2. <https://machinelearningmastery.com/natural-language-processing/>
3. <https://monkeylearn.com/blog/what-is-natural-language-processing/>
4. Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S. Yu, and Lifang He. 2021. *A Survey on Text Classification: From Traditional to Deep Learning.* ACM Trans. Intell. Syst. Technol. 37, 4, Article 111 (April 2021), 39 pages. https://doi.org/10.1145/1122445.1122456
5. <https://machinelearningmastery.com/transfer-learning-for-deep-learning/>
6. Misra, Rishabh and Prahal Arora. "Sarcasm Detection using Hybrid Neural Network." arXiv preprint arXiv:1908.07414 (2019).
7. <https://www.geeksforgeeks.org/removing-stop-words-nltk-python/>
8. <https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html>
9. <https://huggingface.co/docs/transformers/index>
10. <https://towardsdatascience.com/keeping-up-with-the-berts-5b7beb92766>
11. <https://arxiv.org/abs/1810.04805>
12. <https://arxiv.org/abs/1910.01108>
13. https://arxiv.org/abs/1909.11942
14. <https://arxiv.org/abs/2006.03654>

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40.

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance detection with bidirectional conditional encoding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 876–885, Austin, Texas. Association for Computational Linguistics.

James W. Cooley and John W. Tukey. 1965. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90):297–301.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. [Noise reduction and targeted exploration in imitation learning for abstract meaning representation parsing](http://aclweb.org/anthology/P16-1001). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1–11. <https://doi.org/10.18653/v1/P16-1001>.

Dan Gusfield. 1997. *Algorithms on Strings, Trees**and Sequences*. Cambridge University Press, Cambridge, UK.

Mary Harper. 2014. [Learning from 26 languages: Pro- gram management and science in the babel program](http://aclweb.org/anthology/C14-1001). In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, page 1. <http://aclweb.org/anthology/C14-1001>.

Alexander V. Mamishev and Murray Sargent. 2013. *Creating Research and Scientific Documents Using Microsoft Word*. Microsoft Press, Redmond, WA.

Alexander V. Mamishev and Sean D. Williams. 2010. *Technical Writing for Teams: The STREAM Tools Handbook*. Wiley-IEEE Press, Hoboken, NJ.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *Computing Research Repository*, arXiv:1503.06733. Version 2.

1. Appendices

Appendices are added after the References section by restarting the header numbering using style “A, B, C”.

1. <https://www.aclweb.org/portal/content/acl-code-ethics> [↑](#footnote-ref-1)