Fake News Detection

Mahsa Ghaderan

A Thesis

submitted in partial fulfillment of the

requirements for the degree of

Graduate

Iran University of Science and Technology
2021

Reading Committee:
Main Guardian, Chair
First Reader
Second Reader

Program Authorized to Offer Degree:
Computer Science and Engineering

Iran University of Science and Technology

Abstract

Fake News Detection

Mahsa Ghaderan

Chair of the Supervisory Committee:

Professor Sauleh Eetemadi Main Guardian
Computer Science and Engineering

Insert abstract here.

Acknowledgements

I want to give my warmest thanks to my supervisor, professor Sauleh Etemadi, Who made this project possible. He fully supports me and guides me. He also led me in each stage of writing and doing this report.

Specially thanks to my mother, father, and my sister, who support me every single day of my life. They were motivating me in this way.

I am also thankful for my teammates and the friends Mr. Majid Zarharan, Mr. Mehdi Moghadami, Mr. Armin Gholampoor, and Miss Zahra Hosseini. They helped me to improve this project and giving me valuable advice.

Finally, Thank God for letting me through all these hard days and give me such power to finish this work. I keep trusting you for my future.

DEDICATION

To ????

Contents

List of Figures

List of Tables

Chapter 1

Stance Detection

1.1 Introduction

The purpose of stance detection is to automatically find a relation type between specified sentences against a given text. So it is possible to evaluate what is the orientation of a news source towards a particular issue Riedel et al. [2017]. The selected sentence could be a claim, a news item, an idea, a social network post, or any other source. Also, the text could be retrieved from news agencies, weblogs, posts that are shared on social media, and any other available text. Choosing the source and context of sentences and texts depends on the goal of the defined task. Four considered labels are *Agree*, *Disagreed*, *Discussed*, and *Not Enough Information*.

Gathering a sufficient amount of data is a vital step to achieve reliable predictor model. Both number of records in the dataset and the quality of each sample have a significant effect on prediction accuracy. Dulhanty et al. [2019]) gathered fifty thousand articles-headline pairs for their dataset, and achieved a respectable ninety percent accuracy, which was considerably higher than previous attempts by other researchers (Giansiracusa [2021]). Though, in some contexts, there isn't enough available data. Here is where transfer learning methods play a vital role and compensate lack of data. We pre-trained a model on a large text corpus in a general context and then fine-tune the model on task-specified data. Dulhanty et al. [2019] used RoBERTa model which is pre-trained on Facebook data. Then fine-tune it on the specific task at hand. Also, Schiller et al. [2020] improved its accuracy by fine-tuning BERT model.

Researchers have suggested various methods for stance classification. One cluster of methods mainly focuses on deep learning approaches. The precision of models is improved by using word embeddings such as BERT, using recurrent neural networks such as LSTM, BiLSTM (Mrowca et al. [2017]), and utlizing attention-based network (Mrowca et al. [2017]). Besides, novel model architectures such as Memory Network (Mohtarami et al. [2018]), are currently proposed.

One possible way of evaluating the veracity of a given claim is detecting the stance of that claim against available trusted sources. Stance detection task traditionally were used in political and ideological debates fields [Schiller et al., 2020]. The idea of using Stance Detection techniques to analyze news items correctness has become caught researchers attention since 2016. Consistency of a sentence through sources can be retrieved by stance detection methods. Consequently, stance detection can be considered as a subtask of fake news detection (Deepak P [2021]), and It is easier to judge a claim by its stance against other sources. So estimating the stance of a particular claim against available documents can be the first step in detecting fake news.

1.2 Literature Review

Many researchers have done to improve stance detection accuracy. Different ideas and architectures have been applied. In the following part, previous research and papers have been overviewed.

In 2016 Augenstein et al. [2016] started working on the challenging task without assuming neither target is clearly mentioned in the text nor training date is given for every target. Their dataset construct of tweets mostly containing politicians and popular issues. The paper mainly focuses on detecting a stance with respect to unseen targets. Augenstein et al. [2016] used conditional LSTM encoding on Tweeter data. Augenstein et al. [2016] inferred that using unconditional LSTM has the best performance for unseen targets. Their results are improved by using bidirectional encoding. Using LSTM-based models lead to better results than Majority class, SVM (Chang and Lin [2011]), and BoW (Harris [1954]) as baselines in this experiment.

Riedel et al. [2017] has developed an end-to-end stance detection system including lexical and similarity-based features which is passed through a multi-layer perception model. UCLMR's[1] model claimed third

[1] UCL Machine Reading

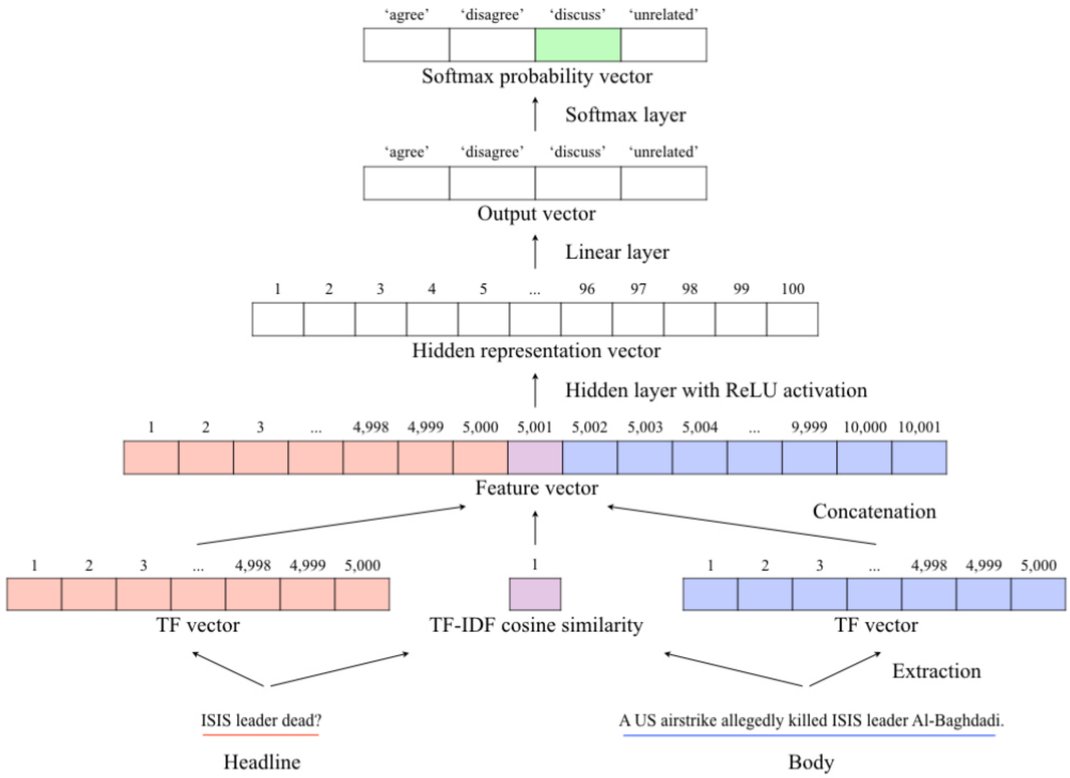


Figure 1.1: Schematic diagram of UCLMRs system.

place in FNC-1[2]. The model architecture is illustrated in Figure 1.1. Headline and body texts are tokenized by scikit-learn[4]. Furthermore, Riedel et al. [2017] used both term frequency (TF) vector of headline and claim and cosine similarity between head and body, and $l2$-normalized TF-IDF vectors. Besides, stop words are excluded. Riedel et al. [2017] achieved FNC-1 score of 75.20% on the test data set.

Sun et al. [2018] have mainly focused on linguistic information such as polarity and argument of each document to represent a document. As shown in figure 1.2 document, sentiment, dependency, and argument representations are used in model architecture. Sun et al. [2018] concluded that every linguistic information with attention mechanism improves stance detection, And using linguistics features altogether outperform using them individually. More details are provided below.

- **Document Representation**: Sun et al. [2018] used a LSTM model to represent each document.

[2]First stage of a competition Fake News Challenge(FNC-1) is exploring how artificial intelligence technologies could be leveraged to combat fake news[3]. Numerous researchers has interest in this field and many papers has published besides this challenge.
[4]F. Pedregosa and Duchesnay. [2011]

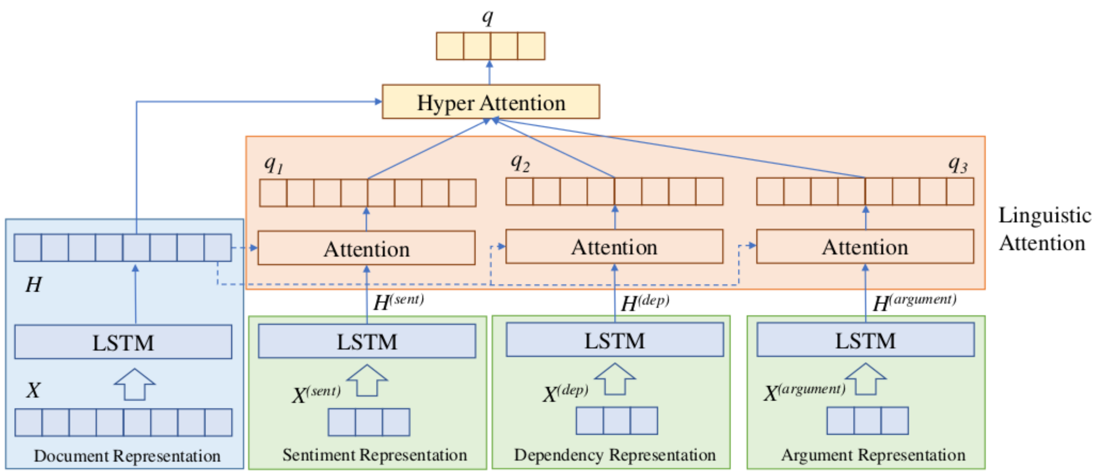


Figure 1.2: Overview of Sun et al. [2018] model.

- **Sentiment Representation**: As sentiment representation, an LSTM model is used to learn the representation of sentiment information. The sentimental word sequence of each document is extracted from sentiment lexicon.

- **Dependency Representation**: This feature is used to capture inter-word relationships. Firstly, relations from the dependency parser are extracted. Then, representation of dependency sequence is learnt by using an LSTM layer.

- **Argument Representation**: The argument is considered as the author's stance. Sun et al. [2018] used a binary classification to detect the document's argument sentence. Then, it learned the sequence representation of word sequence in argument sentences by making use of LSTM layer.

In addition, it utilized a hierarchical attention network in order to weigh the importance of linguistic information, and learn the mutual attention between the document and linguistic information. Sun et al. [2018] mentioned that the Hyper Attention layer in Figure 1.2, had a considerable influence on model performance.

Mohtarami et al. [2018] present a novel end-to-end memory network in 2018 to predict stance and extract snippet of the prediction. The proposed model mainly focuses on relevant paragraphs. This model incorporates recurrent, convolutional neural networks and similarity matrixes. Mohtarami et al. [2018] mentioned that detecting *Disagree* is the hardest label to predict. To overcome the imbalance issue, Mohtarami et al.

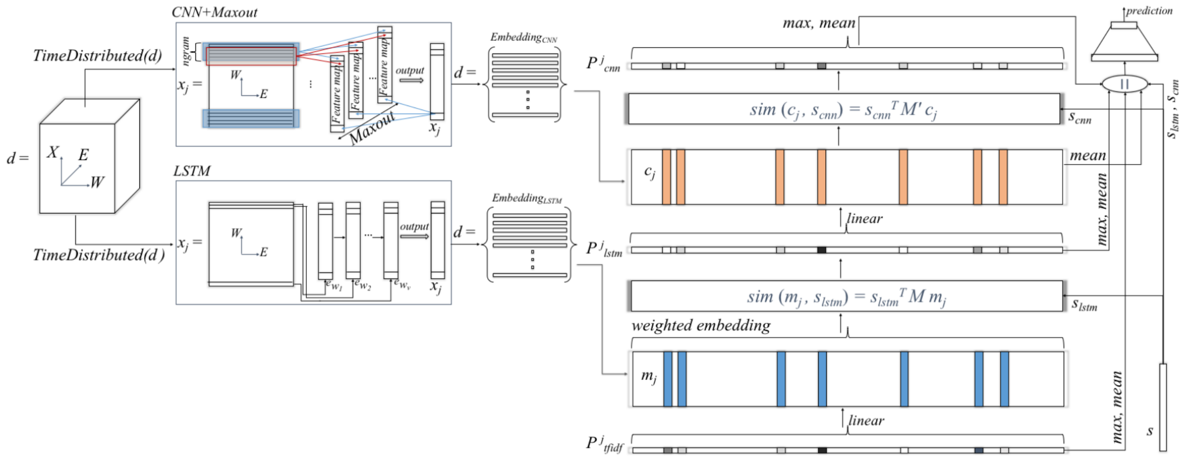


Figure 1.3: Architecture of Memory Network for stance detection.

[2018] select the same number from each class in each iteration.

Schiller et al. [2020] mainly focused on the robustness of a stance detection classifier. Trained models on a single data set in a special domain, won't be robust enough on other domains. So they suggested using a multi-domain dataset or use multi-dataset learning methods to improve model generalization. The model architecture is constructed of fine-tuned BERTDevlin et al. [2019] and a single dense classifier at the top. Schiller et al. [2020] used 5 fixed seed values during training and reported averaged results of trained models. Schiller et al. [2020] concluded that MDL(multi-dataset learning) has a significant impact on increasing model robustness.

1.3 Dataset

The first Persian stance detection dataset (Majid Zarharan [2019]) is used in this project. Majid Zarharan [2019] dataset can be used in stance detection, summarization, and fake news detection tasks. This dataset contains 2124 news articles that cover information about 534 claims. Number of samples in each class is specified in table 1.1. Claims are retrieved from Shayeaat [5] and Fakenews[6] websites. Each sample contains 3 different labels for the stance detection task, containing the article's headline toward the claim, the article's body toward the claim, and the article's headline toward its body. Samples are tagged manually. Four

[5]shayeaat.ir
[6]fakenews.ir

Table 1.1: Class distribution

Label	Agree	Disagree	Unrelated	Discuss
Headline to claim	628	210	932	824
Article to claim	189	374	797	1196

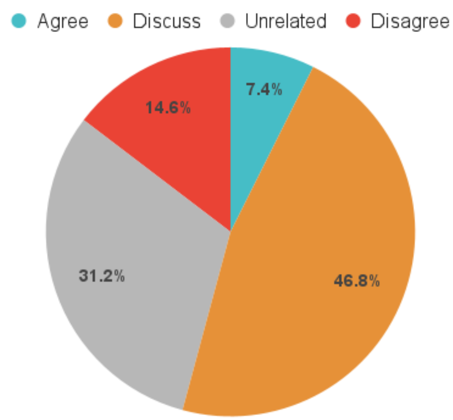
following classes are considered for stance classifying:

- **Agree:** The article clearly states that the claim is True without any ambiguity or amphibology.
- **Disagree:** The article clearly refutes that the claim without any ambiguity or amphibology.
- **Discuss:** The article contains information about the claim but doesn't have any evaluation of its truth.
- **Unrelated:** There isn't any information about the claim in the article.

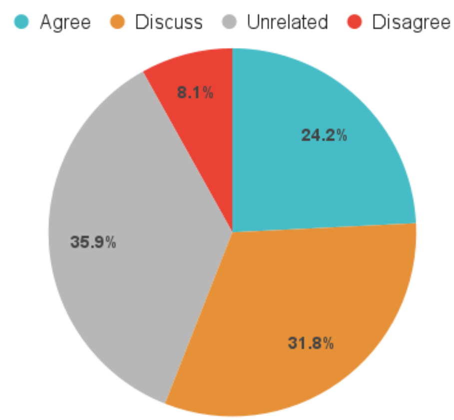
Dataset samples distribution in each class is illustrated in Figure 1.4. According to Figure 1.4, the ratio of *Agree*. Besides, *Disagree* labels are much lower than the others and there is a potential risk for models to be biased on *Unrelated* and *Discuss*. Also, the percentage of *Unrelated* label is higher in headline to claim than article to claim. A headline can be considered as a summary of news body. So, unlike news text, news headline may not have enough information to evaluate a claim. Moreover, ratio of *Discuss* to *Agree* and *Disagree* is higher in Article to claim in comparison to headline to claim. Accordingly, it seems that news agencies choose more controversial headlines to appeal reader's attention.

Furthermore, this dataset covers each claim veracity according to related news articles. In this dataset, the main focus has been on published fake news, this can be inferred from figure 1.5. Three following labels are also considered for classifying news veracity for each claim-headline and claim-body pairs.

- **True16:** Reliable news agencies have asserted that this claim is a fact.
- **False120:** Unreliable news agencies have spread data about this claim and reliable news agencies have considered this claim as hearsay.
- **Unknown5:** There isn't enough integrity between reliable news agencies sources.



(a) Artcile to Claim



(b) Headline to claim

Figure 1.4: Comparison between Article to claim and Headline to claim labels, samples distribution in Majid Zarharan [2019] dataset.

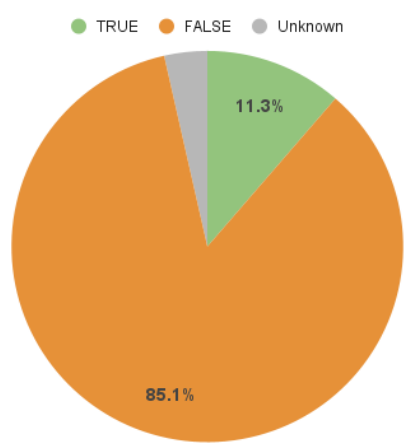


Figure 1.5: Claim veracity label's distribution in Majid Zarharan [2019] dataset.

1.4 Experiments

The headline of news is a summary of its body content and most of the time, it carries valuable data. So, we focused on detecting the news headlines stance towards claim(H2C), as well as the news articles stance towards a claim(B2C). According to the lower amount of text in the news headline, most of the experiments are firstly applied to H2C. Then, better approaches are applied to A2C.

1.4.1 Preprocessing

The first and mandatory preprocessing step is to tokenize words in the corpus in order to remove and detect special words. Four different following tokenizer performance on Persian language has been evaluated.

- **Hazm[7]:** Hazm is a python library Persian language processing tool kit. It has variety of functions such as word and sentence tokenizer, word lemmatizer, POS tagger, Shallow, and Dependency parser.
- **NLTK[8]:** NLTK (Natural Language ToolKit)) is a python platform to work easier with human language data. This tool kit supports more than 50 datasets and supports numerous languages including Persian.
- **Stanford[9]:** Stanford NLP tools is also another advantageous NLP tools package that is currently switched to *Stanza*. It is efficient for linguistic analysis and supports more than 70 human languages and releases new versions constantly.
- **BERT[10]:** BERT (Devlin et al. [2019]) is a transformer-based machine-learning model which is currently used in various natural language processing tasks. Persian pre-trained BERT model[11] can be used for tokenizing corpus too.

It is vital in the tokenizing step to break the corpus into correct words. It may happen that tokenizers generate meaningless words, this is why tokenizers limit the number of accepted words. Also, sometimes tokenizers haven't seen words before, and omit them while tokenizing the corpus. According to stated reasons, it is important to choose a tokenizer carefully.

[7] sobhe.ir
[8] nltk.org
[9] stanfordnlp.github.io/stanza
[10] huggingface.co/transformers/main_classes/tokenizer.html
[11] huggingface.co/HooshvareLab/bert-base-parsbert-uncased

After tokenization, a list of punctuation and Persian stop-words are considered as *denied* and will remove from the corpus in this step. Firstly, the same stop-words was used which have been used by Majid Zarharan [2019]. After reviewing preprocessed corpus, it was hard to infer the stance from text pieces. So we chose stop-words carefully in a way not to lose refuting or supporting expressions. Kharazi[12] has classified Persian stop-words into verbal, nonverbal, and short. Verbs carry valuable information in news. Nonverbal stop-word class is a better choice to remove low-value words in this task. Besides some keywords in news fields are removed from Kharazi's[12] gathered stop-words.

Also English number characters will remove from the corpus before tokenizing. After preprocessing tokens, all tokens will be concatenated with a space character and considered as prepossessed and clean corpus.

1.4.2 Word Representation

To represent a corpus, tokens should be converts to vectors. Good vectors have to carry semantic of each word or n-grams, sequential words contents, and be as brief as possible. As a baseline three different Bag-of-word (Harris [1954]), TF-iDF (Sammut and Webb [2010]), and Word-to-Vector (Tomas Mikolov [2013]) algorithms are evaluated against each other. More details are explained below.

BoW[13]

Bag-of-Words (Harris [1954]) model, is a way to represent texts. BoW keeps words frequency and dismisses word's orders. Dimension of text representation is equal to the number of specified words plus one for words that don't exist in BoW dictionary. Each cell in output text representation stands for a specific word and the value of that cell is equal to the repetition times of that word in the particular text. As an alternative, it is possible to choose n-gram instead of a single word as BoW dictionary. This may improve BoW model performance in order to keep longer expression semantic but on the other hand dimension of representation exceed vastly. One disadvantage of BoW is it doesn't specify any relation between words with similar semantic meanings.

[12] github.com/kharazi/persian-stopwords
[13] en.wikipedia.org/Bag-of-words_model

TF-IDF

Term FrequencyInverse Document Frequency (Sammut and Webb [2010]) is an algorithm to present the importance of each word in a corpus in a statistical way. According to the repetition of a specific word in each document and inverse effect of the number documents that the word appears in, a float number will be assigned to each word in that corpus.

- **Trem frequency (TF):** This item represents how many times a word is used in each document. *TF* should be calculated for each document separately. *TF* term calculate from equation 1.1

$$tf\left(t, D\right) = \frac{f_{t,d}}{\Sigma_{t^{`} \in d} f_{t^{`},d}} \tag{1.1}$$

- **Inverse Document Frequency (iDF):** This term presents how much information a word has. The less repetition of a word through documents, the more information it has. This term is calculated from equation 1.2.

$$idf\left(t, D\right) = \log \frac{N}{\left|\{d \in D, t \in d\}\right|} \tag{1.2}$$

After calculating *TF* and *iDF*, TF-iDF value for each word calculates from equation 1.3.

$$tf - idf\left(t, d, D\right) = tf\left(t, D\right).idf\left(t, D\right) \tag{1.3}$$

W2V[14]

Word2vec (Tomas Mikolov [2013]) is a neural network-based model which learns each word semantic and even it can recommend words with similar meaning to a specific word. Word2Vec represents each word with a vector instead of a number. After the training phase, each word vector serves numbers that are generally similar to words with similar meaning, and words with less correspondence have lower vector similarities. This vector is called word-embedding. It is necessary to have a large corpus in order to train a powerful model which can predict each word vector precisely. Multiple alternative algorithms for Word2Vec exist. In this project, FastText[15] Word2Vec model is used with vector lengths equal to 300 which is trained on Persian

[14] en.wikipedia.org/wiki/Word2vec
[15] fasttext.cc

Wikipedia website. Fasttext is an extension of Word2Vec model which treats each word as concatenated n-grams.

1.4.3 Features

In machine learning algorithms, feature engineering can be considered as the most important step because the desired model trains patterns only corresponding to predictors. Extracting sufficient predictors as compact as possible to having the best predicting accuracy and time efficient , requires numerous experiments. In the following part of this section, extracted features are explained in detail.

Similarity

Similarity score is offered by Majid Zarharan [2019]. This feature calculates how much a claim is similar to a headline or a news article, depends on the task. Three following sequence matching score is considered for this feature by utlizing *difflib*[16] python library.

- Ratio: Similarity score in float range 0,1. This paramets calcualtes from equation 1.4

$$ratio = \frac{2.0 * M}{T} \tag{1.4}$$

  where:

  T — Number of elements in both sequences.

  M — Number of matches.

- Quick Ratio: This parameter estimate an upper bound on Ratio.
- Real Quick Ratio: This parameter estimate an upper bound on Ratio.

Root Distance

This feature is suggested by Majid Zarharan [2019]. Root Distance stands for distance between root of a headline and some collected hedge, refuting and reporting words. Firstly, set of words which considered as mentioned group are gathered and then for each word distance is calculated.

[16] docs.python.org/3/library/difflib.html

ImportantWords

List of controversial and challenging word in news are gathered by Majid Zarharan [2019] and considered as *important-words*. This feature is a zero based list with the length of important words, and each cell stands for one word in *important-words*. List carries number of repetition of desire words in a specific news article.

Is Question

Is-Question identifies whether a claim or headline a news article ends with question marks or not. Majid Zarharan [2019] dataset contains a column dedicated to this feature.

Has Two Parts

Has-Two-Parts is if a claim is construct of two separately parts. Majid Zarharan [2019] dataset contains a column dedicated to this feature.

Polarity

Polarity of a text can be utilized in variety of tasks. This feature presents how positive or negative a text is. Different algorithm are developed to predict polarity of a text. In this project Dashtipour et al. [2016] dataset is used to calculate each sample sentiment. Dashtipour et al. [2016] contains dictionary of words with their sentiment score between -1 and 1. The more negative meaning a word has, the less value it's polarity has. For each word presents in each sample at most first 30 nonzero polarity value saves in a zero initialed vector with 30 length. As Dashtipour et al. [2016] contains only 1500 word polarity values, it can't cover all words in corpus and it has far way to improve.

In this project an idea is applies to extend PerSent (Dashtipour et al. [2016]) polarity dataset is to use a language model. It is possible to predict similar words with a particular word and estimate their similarity score with a language model. Firstly, similar words that don't polarity score in PerSent with their similarity scores extract from a pre-trained language model. Then search each word in PerSent dataset and apply equation 1.5 average through all similar words polarity score, to estimate the desired word polarity score.

$$polarity\_score\,(w) = \frac{\Sigma_{w^{`} \in W} Similatiry\left(w^{`}, w\right).Polarity\left(w^{`}\right)}{\Sigma_{w^{`} \in W} Similatiry\left(w^{`}, w\right)} \tag{1.5}$$

where:

$$w \text{ — Desire word} \notin \text{PerSent datast.}$$

$$W \text{ — Similar words, Predicted by the language model}$$

$$Similarity \text{ — Similarity score for 2 words which is predicted by the language model.}$$

$$polarity \text{ — Polaroty score which is estimated by Dashtipour et al. [2016] dataset.}$$

One alternative is to use deep neural networks model to predict polarity whether word-level or sentence-level. But due to lack of Persian dataset in this context it is not possible. Available datasets for sentiment analysis are mainly gathered from costumer comments on special businesses. For instance Dehkharghani [2019] used 2 different dataset, first it translated English sentiment analysis corpus and second used comments on hotel. Mehrdad Farahani [2020] used dataset from SnappFood[17], DigiKala[18] comments. One main problem with these datasets are difference use of language between users comments and news. Users mostly use everyday language on the other hand news agencies use formal language.

1.4.4 Machine Learning

Machine learning algorithms aim to learn patterns on a corpus of data while training procedure, then predict class of new test data by those patterns (Giansiracusa [2021]). Machine learning algorithms have powerful performance even in complex problems. Machine learning algorithms in compare to deep learning models learn patterns according to its fed manually extracted predictors, and we don't have any other choice rather than relying on those number of extracted predictors (Giansiracusa [2021]). So extracting useful features is a critical step in machine learning. The more meaningful and suitable predictors they see for a task, the better patterns they can find during training procedure. Performance of each predictor described in section 1.4.3, is evaluated by following machine learning methods.

Gaussian Naive Bayes

First machine learning algorithm used to classify stance is Gaussian Naive Bayes Classifier (John and Langley [1995]). It is an alternative to machine learning Naive Bayes classifier that inspires from Bayes Theo-

[17] snappfood.ir
[18] digikala.ir

rem[19]:

$$P\left(y|X\right) = \frac{P(X|y).P(X)}{P(y)}$$

where:

X — List of predictors that are independent to each other.

y — Label of a class.

$P\left(X|y\right)$ — Probability of class with label y from given X predictors.

Naive Bayes Classifier algorithm estimates probability of each class. Gaussian Naive Bayes means that predictors are continuous and follow Gaussian distribution:

$$p\left(x = v|C_k\right) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(\epsilon-\mu_k)^2}{2\mu_k^2}}$$

C_k — Class k.

μ_k — Mean of x values associated with C_k

σ_k^2 — Bessel corrected variance x values associated with C_k

SVC[20]

SVC stands for SVM classifier. Support Vector Machines (Chang and Lin [2011]) are group of supervised machine learning models. One of their application is classification. SVM algorithms map each sample to a space that each class samples be as far as possible from other class samples. In the other word SVM id looking for a n-dimensional hyperplane which can separate classes as clearly as possible.

SVC model from *scikit-learn*[21] python library is used in this project. Regularization parameter (c) which is stands for strength of regularization is set to 10. As a kernel three different *rbf*, *sigmoid*, and *poly* hyperplane are evaluated. Kernel specifies the type of separator that SVM algorithm use to distinguish different classes. Furthermore, *class_weight* parameter set to *balanced* which mean set different weight for each class during training to compensate imbalanced data.

[19]en.wikipedia.org/wiki/Naive_Bayes_classifier
[20]wikipedia.org/Support-vector_machine
[21]scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

LinearSVC

LinearSVC is a ulternative algorithm for SVC in large datasets. LinearSVC linearly separates samples. Depends on dataset it may works better than nonlinear SVC. This model is same as SVC from previous part, only different is to set *kernel* parameter equal to linear.

Random Forest

Random Forest (**?**) is a machine learning algorithm to deal with complex classification problems. It construct of many decision trees. The point is that it is more robust than decision threes. Besides Random Forest classifier doesn't need parameter tuning. Each decision three has it's own prediction and final prediction of the model is calculated bt majority voting of each tree output.

In this project, implemented Random Forest algorithm from *scikit-learn*[22] python library is used. Two parameters of *max_depth* and *min_samples_leaf* control size of each decision tree. They are respectively set to 10 and 1. *min_samples_split* bounds least number of samples to apply on a tree, is set to 3.

Logistic Regression

Logistic Regression is a machine learning classification algorithm. It's functionality is mainly for Binary Classification and In multi-class Logistic regression classifies, one class vs rest. This algorithm uses 'Sigmoid function' as it's cost function and its prediction is based on probabilities.

Implemented Logistic Regression algorithm from *sckiti-learn*[23] python library is used. *penalty* parameter could be choosed from *l1*, *l2*, and *elasticnet* for penalization and regularization.

- $l1$

$$min_{\omega,c}\ \frac{1}{2}\omega^T\omega + C\Sigma_{i=1}^{n} \log\left(\exp\left(-y_i\left(X_i^T\omega + c\right)\right) + 1\right)$$

- $l2$

$$min_{\omega,c}\ ||\omega||_1 + C\Sigma_{i=1}^{n} \log\left(\exp\left(-y_i\left(X_i^T\omega + c\right)\right) + 1\right)$$

[22] scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[23] scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

- $elastic - net$

$$min_{\omega,c} \frac{1-\rho}{2}\omega^T\omega + \rho \left|\left|\omega\right|\right|_1 + C\Sigma_{i=1}^{n} \log\left(\exp\left(-y_i\left(X_i^T\omega + c\right)\right) + 1\right)$$

where:

ρ — Controls $l1$ strength (*l1_ration* parameter).

y_i — Takes value between -1, 1.

Elastic-Net Penalization is used with ρ parameter equals to 0.5, means $l1$ and $l2$ have same powers and *solver* parameter which stand for optimizer algorithm is set to *sega* which is an alternative for Stochastic Average Gradient (sag) optimizer.

Another configuration for Logistic Regression is also evaluated. In this setup, optimizer is set to *lbfgs* which performs more robust in larger datasets. Although, *lbfgs* optimizer is slower than *saga*. Also, penalization is set to $l2$ [24].

1.4.5 Balancing

As mentioned in section Dataset, Figure 1.4, number of samples in dataset classes was imbalanced. As a result, models bias on majority class and there may not enough sample in minority class for model to learn that, this leads to having high accuracy score (Equation 1.6) while having low f1 score (Equation 1.7).

$$F1 = \frac{TP + TN}{TP + FN + TN + FP} \tag{1.6}$$

where:

TP — True Positive

TN — True Negative

FP — False Positive

FN — False Negative

[24] scikit-learn.org/stable/modules/linear_model.html#logistic-regression

$$F1 = \frac{2 \times precision \times recall}{precission + recall} \tag{1.7}$$

where:

$precission$ — $precision = \frac{TP}{TP+FP}$

$recall$ — $recall = \frac{TP}{TP+FN}$

There are several algorithm to deal with imbalanced datasets. In Majid Zarharan [2019] minority class forms only 7.4% of data (Figure 1.4). So it's not practical to rely on only one method and except to perform in the best way. Consecuently, three different methods used in this project in order to dealing with this phenomenon. Methods of balancing dataset which are used in this project is described in following sections.

Extending dataset:

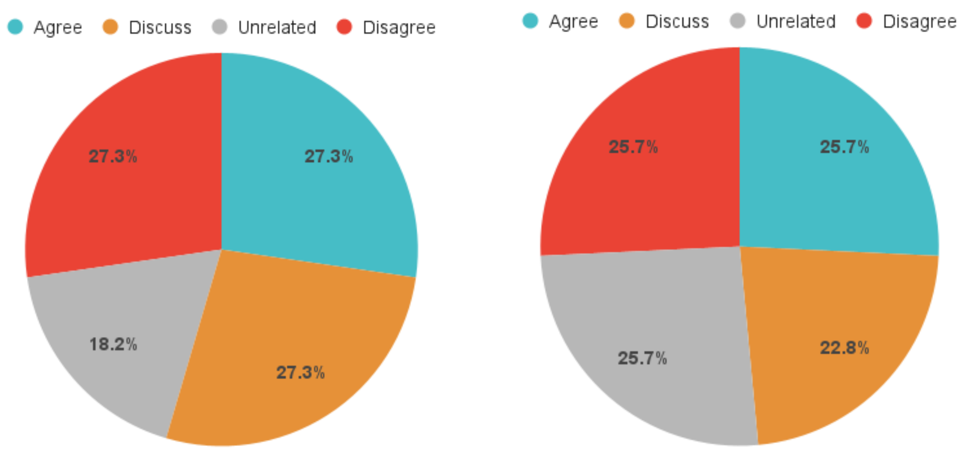
The simplest method is to gather data for classes except majority class. But unfortunately it is not always practicable. Another way of extending dataset is to use another existing dataset which has similar gathering logics and it is possible to map these two dataset classes.

ParsFEVER (Zarharan et al. [2021]) is a persian dataset set based on FEVER (Thorne et al. [2018]) dataset is gathered for fact extraction and verification task. Zarharan et al. [2021] claims are generated from Wikipedia[25] articles manually, then evidences for each claim are extracted from Wikipedia separately by distinct annotators. This dataset contains three *Support*, *Refute*, and *Not Enough Info* classes.

- **Support:** The article obviously proves the given claim.
- **Refute:** The article obviously disproves the given claim.
- **Not Enough Info:** There isn't enough information in the article about the claim.

According to Figure 1.4 two *Agree* and *Disagree* class in Majid Zarharan [2019] dataset suffers from lack of samples. In this project, *Supports* and *Refutes* samples from Zarharan et al. [2021] dataset are mapped to *Agree* and *Disagree* class of Majid Zarharan [2019] dataset respectively. But it is not possible to extend *Discuss* or *Unrelated* class by ParsFEVER, because they are both merged in *Not Enough Info*

[25] wikipedia.org



(a) Artcile to Claim

(b) Headline to claim

Figure 1.6: Comparison between Article to claim and Headline to claim labels, samples distribution in Majid Zarharan [2019] dataset, after extending by Zarharan et al. [2021] .

class. As a result, two *Agree* and *Disagree* extended as much as possible with random selected samples from ParsFEVER dataset. Sample distribution is illustrated in figure 1.6. Dataset is still imbalanced in one class for both Article to Claim and Headline to Claim.

Oversampling and Undersampling:

Another common way of dealing with imbalanced dataset is automatically augmenting samples to achieve balanced class distribution (Oversampling) or even reduce number of samples in majority class (Undersampling). Undersampling is applicable on large dataset. But in small dataset it's not wisely to ignore some samples. Oversampling methods suites for such datasets. According figure 1.6 Despite extending dataset with Zarharan et al. [2021] dataset, balancing dataset is still needed. Though, oversamping shoud be performed on minority class only. It is important to split test and train sets before resampling, and oversampling should be only apply on train set. Resampling methods that evaluated are:

- **RandomOverSampler[26]:** This method randomly peak samples from classes and resample them. Random Over Sampler is the most naive algorithm and its performance is same as increasing minority class loss weight. Another variant of this method is smoothed bootstrap oversampling. It is generally similar to Random Oversampler, but new samples don't exactly overlap original samples.

[26]imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.RandomOverSampler.html

They are adjacent to source samples. This variant can be implement by *shrinking* parameter in *RandomOverSampler* from *imblearn* python library.

- **SMOTE:** Synthetic Minority Over-sampling Technique (Chawla et al. [2002]) is an oversampling method by generating new samples by interpolation. It's not important for smothe sampler that which point is choosed to be resampled.

- **SVMSMOT[27]:** SVMSMOTE (Wu and Chang [2003]) is a variant of SMOTE oversampler which uses SVM (Section 1.4.4) algorithm to choose resampling samples. One strength of this model that it is effective to both vector data and sequence data (Wu and Chang [2003]).

- **BorderlineSMOTE[28]:** BorderlineSMOTE (Han et al. [2005]) is also another variant of SMOTE oversampler. Borderline samples are mainly chosen to get resample in this variant. Han et al. [2005] has achieved better accuracy than SMOTE.

- **ADASYN[29]:** Adaptive Synthetic Sampling Approach for Imbalanced Learning (He et al. [2008]) focuses on generating new samples adjacent to those samples wrongly classified by employing K-Nearest Neighbors classifier. These samples are considered as hard samples because it's not easy for models to predict them, as a result ADASYN increases robustness of desired dataset. This method performs better in compare to Decision Tree and SMOTE algorithms (He et al. [2008]). In this project number of nearest neighbors to generate new sample is set to 9.

All mentioned oversampling methods are evaluated against each other in this project and utilized from oversampling package of *imblearn* [30] python library.

Tune mode parameters:

The last but not least important balancing method is to choose a robust learning algorithm to imbalanced dataset, Choosing weight of each class according to ratio of each class samples and choosing optimizer and

[27]imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SVMSMOTE.html
[28]imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.BorderlineSMOTE.html
[29]imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.ADASYN.html
[30]imbalanced-learn.org/stable/references/over_sampling.html

loss function that can overcome imbalanced dataset. After applying previous methods to balance dataset, this step can be skipped in this project.

1.4.6 Deep Learning

At baseline, machine learning algorithm applied to evaluate features Affects

Schiller et al. [2020]ML models trained on a single dataset usually generalize poorly to other domains.

baseline

Bert

roberta

1.5 Results

Moreover, the black-box nature of machine learning algorithms means that nobody really knows why an AI lie detection system works as it does, nor what it is actually doing.(Giansiracusa [2021])

train procedure for each method

compare test results

1.6 Conclusion

talk about best method

talk about other possible ways

Chapter 2

Fake News

2.1 Literature Review

2.2 Introduction

2.3 Experiments

Ideas and works inorder to detect fake news.

2.4 Results

2.5 Conclusion

Chapter 3

Conclusion

Insert conclusion here. + Future works and ideas to continue this project.

Bibliography

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance detection with bidirectional conditional encoding. In *EMNLP*.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2.

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. 2002. Smote: Synthetic minority over-sampling technique. *Journal Of Artificial Intelligence Research*, 16:321–357.

Kia Dashtipour, Amir Hussain, Qiang Zhou, Alexander Gelbukh, Ahmad Hawalah, and Erik Cambria. 2016. Persent: A freely available persian sentiment lexicon. volume 10023.

Cheng Long Santhosh Kumar G Deepak P, Tanmoy Chakraborty. 2021. Data science for fake news. 42.

Rahim Dehkharghani. 2019. Sentifars: A persian polarity lexicon for sentiment analysis. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 19.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1 (Long and Short Papers), pages 4171–4186. Association for Computational Linguistics.

Chris Dulhanty, Jason Deglint, Ibrahim Ben Daya, and Alexander Wong. 2019. Taking a stance on fake news: Towards automatic disinformation assessment via deep bidirectional transformer language models for stance detection.

A. Gramfort V. Michel B. Thirion O. Grisel M. Blondel P. Pret-tenhofer R. Weiss V. Dubourg J. Vanderplas A. Passos D. Cournapeau M. Brucher F. Pedregosa, G. Varoquaux and M. P. E. Duchesnay. 2011. Scikit-learn: Machine learning in python.

Noah Giansiracusa. 2021. How algorithms create and prevent fake news.

Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. 2005. Borderline-smote: A new over-sampling method in imbalanced data sets learning. *Advances in Intelligent Computing. ICIC 2005*, 3644.

Zellig S. Harris. 1954. Distributional structure. *WORD*, 10(2-3):146–162.

Haibo He, Yang Bai, Edwardo A. Garcia, and Shutao Li. 2008. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1322–1328.

George H. John and Pat Langley. 1995. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, page 338345. Morgan Kaufmann Publishers Inc.

Fatemeh Sadat Rezvaninejad Mahdi Lotfi Bidhendi Shaghayegh Sadat Jalali Sauleh Eetemadi Mohammad Taher Pilehvar Behrouz Minaei-Bidgoli Majid Zarharan, Samane Ahangar. 2019. Persian stance classification dataset.

Marzieh Farahani Mohammad Manthouri Mehrdad Farahani, Mohammad Gharachorloo. 2020. Parsbert: Transformer-based model for persian language understanding. *ArXiv*, abs/2005.12515.

Mitra Mohtarami, Ramy Baly, James Glass, Preslav Nakov, Lluís Màrquez, and Alessandro Moschitti. 2018. Automatic stance detection using end-to-end memory networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 767–776. Association for Computational Linguistics.

Damian Mrowca, Elias Wang, and Atli Kosson. 2017. Stance detection for fake news identification.

B. Riedel, Isabelle Augenstein, Georgios P. Spithourakis, and S. Riedel. 2017. A simple but tough-to-beat baseline for the fake news challenge stance detection task. *ArXiv*, abs/1707.03264.

Claude Sammut and Geoffrey I. Webb, editors. 2010. *TF–IDF*, pages 986–987. Springer US.

Benjamin Schiller, Johannes Daxenberger, and Iryna Gurevych. 2020. Stance detection benchmark: How robust is your stance detection? *ArXiv*, abs/2001.01565.

Qingying Sun, Z. Wang, Qiaoming Zhu, and Guodong Zhou. 2018. Stance detection with hierarchical attention network. In *COLING*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819. Association for Computational Linguistics.

Greg Corrado Jeffrey Dean Tomas Mikolov, Kai Chen. 2013. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, pages 4171–4186. Association for Computational Linguistics.

Gang Wu and E. Chang. 2003. Class-boundary alignment for imbalanced dataset learning.

Majid Zarharan, Mahsa Ghaderan, Amin Pourdabiri, Zahra Sayedi, Behrouz Minaei-Bidgoli, Sauleh Eetemadi, and Mohammad Taher Pilehvar. 2021. Parsfever: a dataset for farsi fact extraction and verification. Association for Computational Linguistics.

Chapter A

Appendix One

A.1 Appendix section 1

Table A.1: Table in the Appendix