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

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

[1]en.wikipedia.org/Bag-of-words_model
[2]en.wikipedia.org/wiki/Word2vec

[3]wikipedia.org/Support-vector_machine

List of Figures

List of Tables

Chapter 1

Introduction

These days, fake news and false information have been vastly flooding the Internet by numerous not verified news agencies with aims ranging from affecting individual peoples beliefs and decisions to influencing major events such as political elections (Mohtarami et al. [2018]). As days go by, the importance of detecting stance automatically is appealing more attention. Insofar as it has been suggested that automatic stance detection is the first step towards assisting human fact checkers to detect inaccurate claims Riedel et al. [2017].

The purpose of automatic stance detection is to find the type of a relation between specified sentences against a given text by utilizing machines. 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.

Chapter 2

Background

TODO

2.1 BERT

Pretrained language models have significant improve on many natural language model tasks such as natural language inference task (Devlin et al. [2019]). One of most impressive and powerful deep learning language model for text processing, is BERT which is stands for Bidirectional Encoder Representations from Transformers (Devlin et al. [2019]) and developed by Google. And has received state of the art result due its previous researches (Devlin et al. [2019]). Singhal et al. [2019] improved its performance by make usage of the BERT language model in fake news detection. BERT base version contains 12 transformer blocks, containing 110 million parameters. There are also another variant of BERT such as large including 24 transformer blocks and multilingual BERT. BERT models are trained of Wikipedia corpus.

One application of BERT is using first top layers of BERT model as contextual word embedding (Deepak P [2021]). This model suggests a vector that representing a word which best fit in the current context. BERT is a masked language model. It randomly masks some tokens and learn to predict masked tokens correctly. BERT utilizes bidirectional transformer, it means that BERT can inferred both left to right and right to left.

TODO

2.2 ALBERT

TODO

2.3 ParsBERT

TODO

Chapter 3

Related Works

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 place in FNC-1[2]. The model architecture is illustrated in Figure 3.1. 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

[1]UCL Machine Reading

[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]

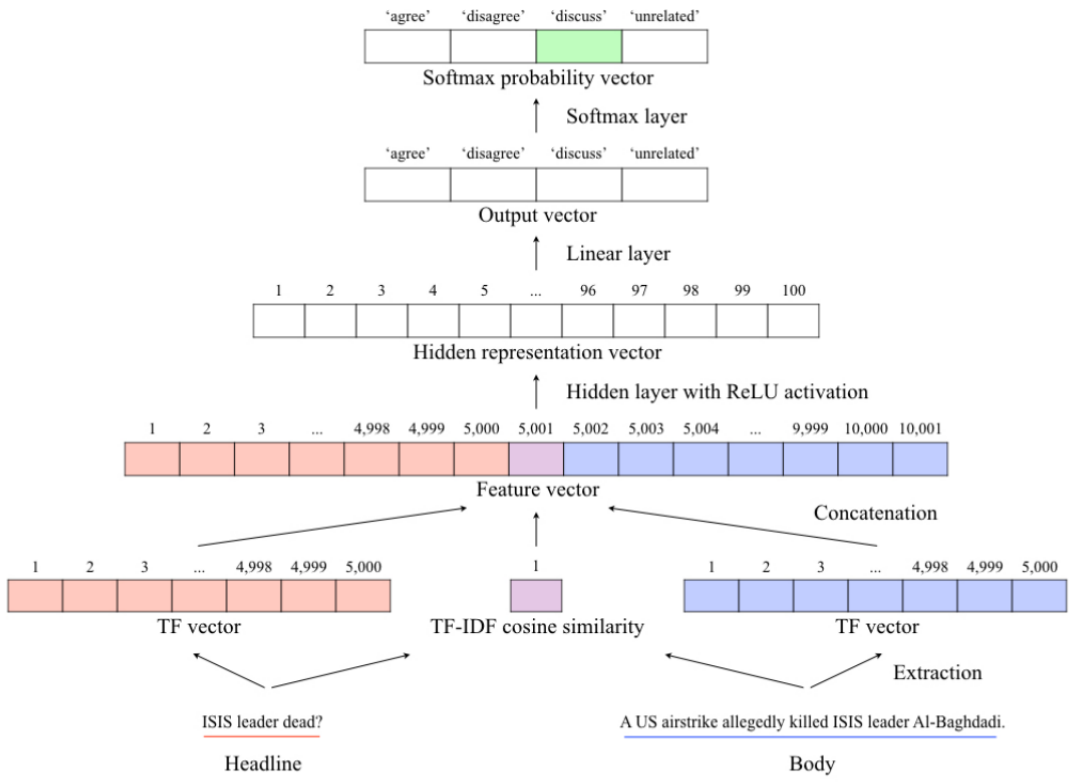


Figure 3.1: Schematic diagram of UCLMRs system.

document to represent a document. As shown in figure 3.2 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.
- **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 3.2, 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. [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

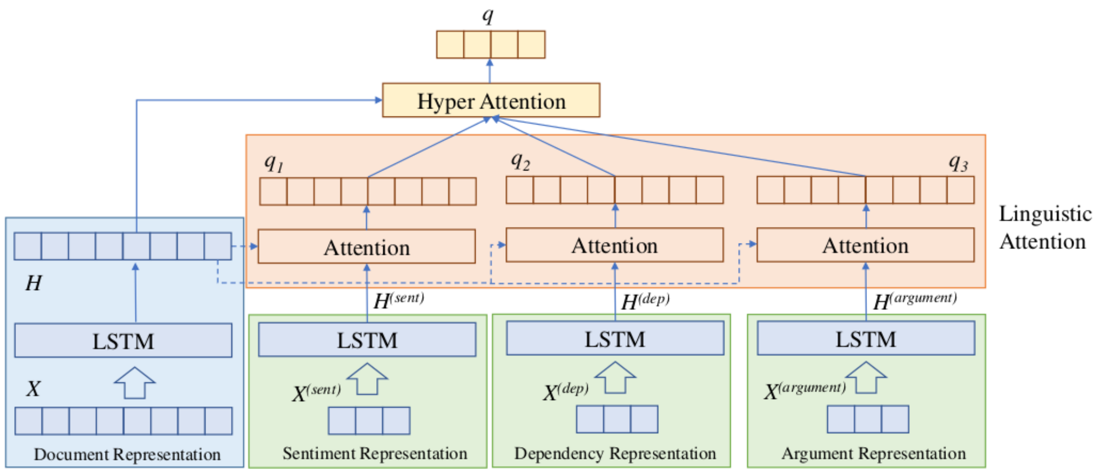


Figure 3.2: Overview of Sun et al. [2018] model.

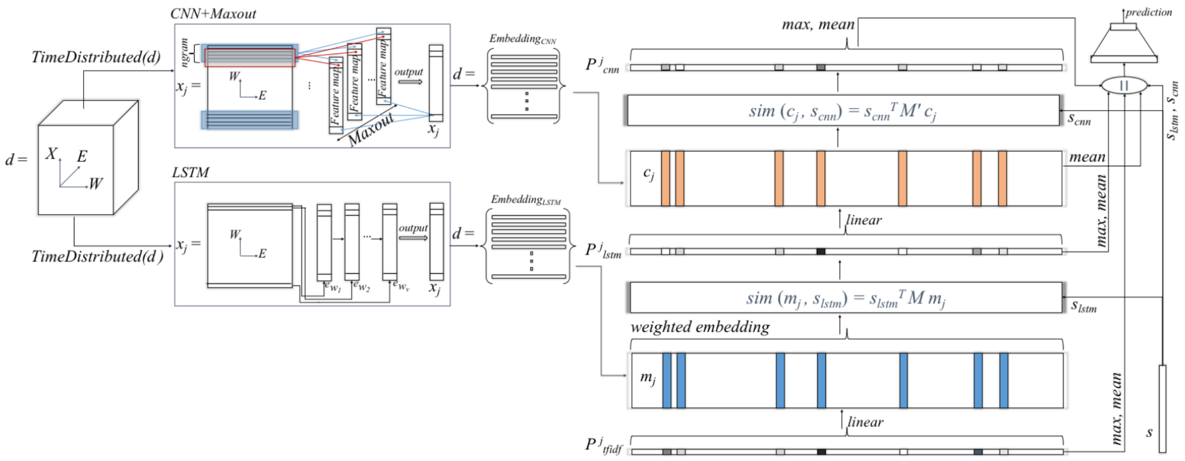


Figure 3.3: Architecture of Memory Network for stance detection.

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.

Chapter 4

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 4.1. Claims are retrieved from Shayeaat [1] and Fakenews[2] 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 following classes are considered for stance classifying:

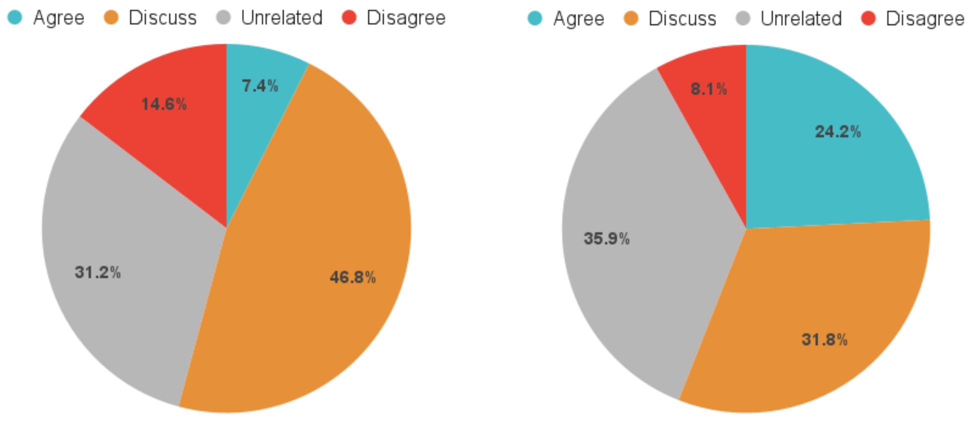
- **Agree:** The article clearly states that the claim is True without any ambiguity or amphibology.
- **Disagree:** The article clearly refutes that the claim without any ambiguity or amphibology.
- **Discuss:** The article contains information about the claim but doesn't have any evaluation of its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

Dataset samples distribution in each class is illustrated in Figure 4.1. According to Figure 4.1, 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

[1]shayeaat.ir
[2]fakenews.ir

Table 4.1: Stance class distribution

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



(a) Artcile to Claim **(b)** Headline to claim

Figure 4.1: Comparison between Article to claim and Headline to claim labels, samples distribution in Majid Zarharan [2019] dataset.

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. Veracity labels statistics (FakeNews Dataset) is illustrated in table 4.2. In this dataset, the main focus has been on published fake news, this can be inferred from figure 4.2. Three following labels are also considered for classifying news veracity for each claim-headline and claim-body pairs.

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

Table 4.2: Fake news data set statistics

Used case	**True**	**False**	**Unknown**
Test set	20	250	8
Training set	91	1003	35
Overall	111	1253	43

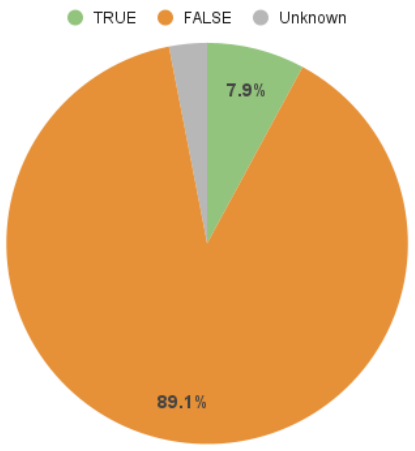


Figure 4.2: Claim veracity label's distribution in Majid Zarharan [2019] dataset.

Chapter 5

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.

5.1 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[1]:** 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. *Hazm* tokenizer is *NLTK* compatible[1].

- **NLTK[2]:** 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[3]:** 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

[1]sobhe.ir
[2]nltk.org
[3]stanfordnlp.github.io/stanza

and releases new versions constantly.

- **BERT[4]:** 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[5] 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.

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[6] 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 adding and removing some words in news fields are evaluated against Kharazi's[6] 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.

5.2 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.

[4]huggingface.co/transformers/main_classes/tokenizer.html
[5]huggingface.co/HooshvareLab/bert-base-parsbert-uncased
[6]github.com/kharazi/persian-stopwords

5.2.1 BoW[7]

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.

5.2.2 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 5.1

$$tf\left(t,D\right)=\frac{f_{t,d}}{\Sigma_{t^{`}\in d}f_{t^{`},d}} \tag{5.1}$$

- **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 5.2.

$$idf\left(t,D\right)=\log\frac{N}{\left|\{d\in D,t\in d\}\right|} \tag{5.2}$$

After calculating *TF* and *iDF*, TF-iDF value for each word calculates from equation 5.3.

$$tf-idf\left(t,d,D\right)=tf\left(t,D\right).idf\left(t,D\right) \tag{5.3}$$

[7]en.wikipedia.org/Bag-of-words_model

5.2.3 W2V[8]

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[9] Word2Vec model is used with vector lengths equal to 300 which is trained on Persian Wikipedia website. Fasttext is an extension of Word2Vec model which treats each word as concatenated n-grams.

5.3 Features

In machine learning algorithms, feature engineering can be considered 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 trials and errors. In the following part of this section, extracted features are explained in detail.

5.3.1 Similarity

The 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 utilizing *difflib*[10] python library.

- Ratio: Similarity score in float range 0,1. This parameters calculates from equation 5.4

$$ratio = \frac{2.0 * M}{T} \tag{5.4}$$

where:

[8]en.wikipedia.org/wiki/Word2vec
[9]fasttext.cc
[10]docs.python.org/3/library/difflib.html

T — Number of elements in both sequences.

M — Number of matches.

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

5.3.2 Root Distance

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

5.3.3 ImportantWords

List of a controversial and challenging words in news 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*. The list carries the number of repetitions of desire words in a specific news article.

5.3.4 Is Question

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

5.3.5 Has Two Parts

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

5.3.6 Polarity

The polarity of a text can be utilized in a variety of tasks. This feature presents how positive or negative a text is. Different algorithms are developed to predict the polarity of a text. In this project, Dashtipour et al.

[2016] dataset is used to calculate each sample sentiment. Dashtipour et al. [2016] contains a dictionary of words with their sentiment score between -1 and 1. The more negative the word meaning, the lower its polarity point. For each word presents in each sample at most first 30 nonzero polarity value saves in a zero initialed vector with 30 lengths. As Dashtipour et al. [2016] contains only 1500 word polarity values, it can't cover all words in corpus and it has a far way to improve.

In this project, an idea is applied 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 the PerSent dataset (Dashtipour et al. [2016]) and apply equation 5.5 average through all similar words polarity scores, to estimate the desired word polarity score.

$$polarity\_score\left(w\right) = \frac{\Sigma_{w`\in W} Similatiry\left(w`, w\right).Polarity\left(w`\right)}{\Sigma_{w`\in W} Similatiry\left(w`, w\right)} \tag{5.5}$$

where:

w	— Desire word $\notin$ PerSent datast.
W	— Similar words, Predicted by the language model
$Similarity$	— Similarity score for 2 words which is predicted by the language model.
$polarity$	— Polaroty score which is estimated by Dashtipour et al. [2016] dataset.

One alternative is to use a deep neural networks model to predict score polarity of a word whether word-level or sentence-level. But due to the lack of a Persian dataset in news context, it is not practical. Available datasets for sentiment analysis are mainly gathered from customer comments on special businesses. For instance Dehkharghani [2019] used 2 different datasets, first it translated English sentiment analysis corpus and second used comments on hotels. Mehrdad Farahani [2020] used dataset from SnappFood[11], DigiKala[12] comments. One main problem with these datasets is the different use of language between user comments and news. Users mostly use everyday language on the other hand news agencies use formal language.

[11]snappfood.ir
[12]digikala.ir

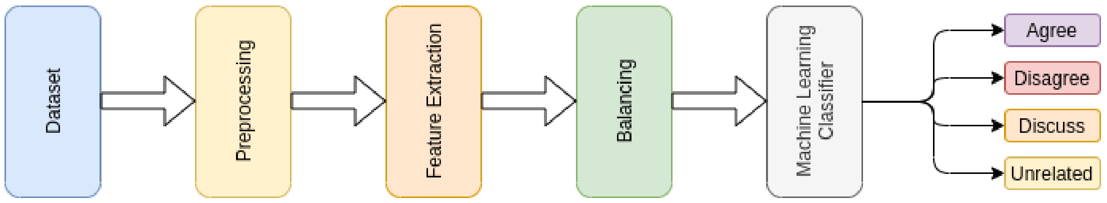


Figure 5.1: Schematic of each machine learning model.

5.4 Machine Learning

Machine learning algorithms aim to learn patterns on a corpus of data while training procedure, then predict classes of news articles test data by those patterns (Giansiracusa [2021]). Machine learning algorithms have powerful performance even in complex problems. In comparison to deep learning models, Machine learning algorithms learn patterns according to their 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 the training procedure. Figure 5.1 illustrates a basic schematic of each machine learning models. The description of each predictor described in detail at section 5.3, is evaluated by following machine learning methods.

5.4.1 Gaussian Naive Bayes

The 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 is inspired by Bayes Theorem[13]:

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

where:

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

y — Label of a class.

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

[13]en.wikipedia.org/wiki/Naive_Bayes_classifier

Naive Bayes Classifier algorithm estimates the 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

5.4.2 SVC[14]

SVC stands for SVM Classifier. Support Vector Machines (Chang and Lin [2011]) are a group of supervised machine learning models. One of their applications is the classification problem. SVM algorithms map each sample to a space such that samples of a particular class locate as far as possible from other classes samples. In other words, SVM is looking for an n-dimensional hyperplane which can separate classes as clearly as possible.

SVC model from *scikit-learn*[15] python library is used in this project. Regularization parameter (c) which is stands for strength of regularization is tuned depends on task. As the kernel three different *rbf*, *sigmoid*, and *poly* hyperplane are evaluated. Kernel specifies the type of separator that SVM algorithm uses 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.

5.4.3 LinearSVC

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

[14]wikipedia.org/Support-vector_machine

[15]scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

5.4.4 Random Forest

Random Forest (Liaw and Wiener [2002]) is a machine learning algorithm to deal with complex classification problems. It constructs 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 its own prediction and the final prediction of the model is calculated bt majority voting of each tree output.

In this project, implemented Random Forest algorithm from *scikit-learn*[16] python library is used. Three parameters of *max_features* (Maximum number of features allowed to use for each tree), *estimator* (Number of decision trees), and *criterion* (Algorithm to measure quality of splits in nodes) are tuned for the desired task.

5.4.5 Logistic Regression

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

Implemented Logistic Regression algorithm from *sckiti-learn*[17] python library is used. *penalty* parameter could be chosen from *l1* (Equation 5.6), *l2* (Equation 5.7), and *elasticnet* (Equation 5.8) for penalization and regularization.

- $l1$

$$min_{\omega,c}\ \frac{1}{2}\omega^T\omega + C\Sigma_{i=1}^{n}\log\left(\exp\left(-y_i\left(X_i^T\omega + c\right)\right)+1\right) \tag{5.6}$$

- $l2$

$$min_{\omega,c}\ ||\omega||_1 + C\Sigma_{i=1}^{n}\log\left(\exp\left(-y_i\left(X_i^T\omega + c\right)\right)+1\right) \tag{5.7}$$

- $elastic - net$

$$min_{\omega,c}\ \frac{1-\rho}{2}\omega^T\omega + \rho\,||\omega||_1 + C\Sigma_{i=1}^{n}\log\left(\exp\left(-y_i\left(X_i^T\omega + c\right)\right)+1\right) \tag{5.8}$$

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

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 the same powers and *solver* parameter which stands for optimizer algorithm is set to *sega* which is an alternative for Stochastic Average Gradient (sag) optimizer.

Another optimization algorithm for Logistic Regression such as *sag*, *lbfgs*, and *liblinear* are also evaluated. In this setup, penalization is set to $l2$. *lbfgs* optimizer performs more robust in larger datasets. Although, it is slower than *saga*[18].

5.5 Balancing

As mentioned in section Dataset, Figure 4.1, the 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 a model to learn that, this leads to having high accuracy score (Equation 5.9) while having low f1 score (Equation 5.10).

$$F1 = \frac{TP + TN}{TP + FN + TN + FP} \tag{5.9}$$

where:

TP — True Positive

TN — True Negative

FP — False Positive

FN — False Negative

$$F1 = \frac{2 \times precision \times recall}{precission + recall} \tag{5.10}$$

where:

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

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

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

There are several algorithms to deal with imbalanced datasets. In Majid Zarharan [2019] minority class forms only 7.4% of data (Figure 4.1). So it's not practical to rely on only one method and except to perform in the best way. Consequently, three different methods were used in this project in order to deal with this phenomenon. Methods of balancing a dataset which is used in this project are described in the following sections.

5.5.1 Extending dataset

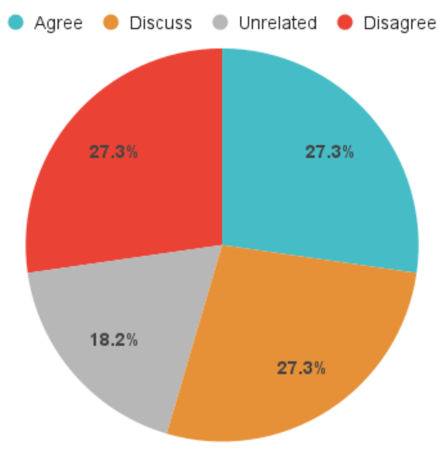
The simplest method is to gather data for classes except for the majority class. But unfortunately, it is not always practicable. Another way of extending a dataset is to use another existing dataset which has similar gathering logic and it is possible to map these two dataset classes.

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[19] articles manually, then pieces of evidence for each claim are extracted from Wikipedia separately by distinct annotators. This dataset contains three *Support*, *Refute*, and *Not Enough Info* classes.

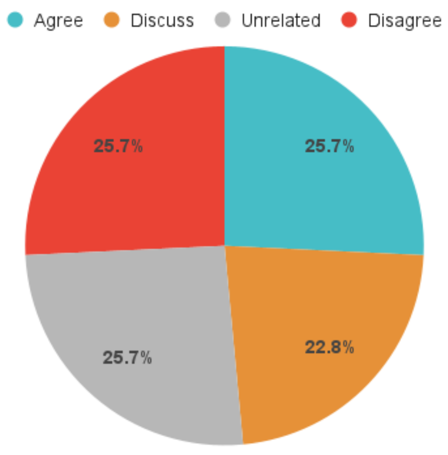
- **Support:** The article obviously proves the given claim.
- **Refute:** The article obviously disproves the given claim.
- **Not Enough Info:** There isn't enough information in the article about the claim.

According to Figure 4.1 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* 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 5.2. Dataset is still imbalanced in one class for both Article to Claim and Headline to Claim.

[19] wikipedia.org



(a) Artcile to Claim



(b) Headline to claim

Figure 5.2: Comparison between Article to claim and Headline to claim labels, samples distribution in Majid Zarharan [2019] dataset, after extending by Zarharan et al. [2021] .

5.5.2 Oversampling and Undersampling

Another common way of dealing with an imbalanced dataset is automatically augmenting samples to achieve balanced class distribution (Oversampling) or even reduce the number of samples in the majority class (Undersampling). Undersampling is applicable on a large dataset. But in small datasets, it's not wise to ignore some samples. Oversampling methods suites for such datasets. According to figure 4.1, oversampling should be performed classes except the majority class. It is important to split test and train sets before resampling, and oversampling should be only apply on the train set. Resampling methods that were evaluated are:

- **RandomOverSampler[20]:** 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. 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

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

method by generating new samples by interpolation. It's not important for smothe sampler that point is chosen to be resampled.

- **SVMSMOT[21]:** SVMSMOTE (Wu and Chang [2003]) is a variant of SMOTE oversampling method which uses SVM (Section 5.4.2) algorithm to choose resampling samples. One strength of this model is that it is effective for both vector data and sequence data (Wu and Chang [2003]).

- **BorderlineSMOTE[22]:** 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[23]:** 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 the robustness of the desired dataset. This method performs better in comparison to Decision Tree and SMOTE algorithms (He et al. [2008]). In this project number of nearest neighbors to generate a 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* [24] python library.

5.5.3 Tune mode parameters

The last but not least important balancing method is to choose a robust learning algorithm for an imbalanced dataset, Choosing a weight of each class according to the ratio of samples of each class and choosing an optimizer and loss function that can overcome an imbalanced dataset. After applying previous methods to balance the dataset, this step can be skipped in this project.

[21]imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SVMSMOTE.html
[22]imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.BorderlineSMOTE.html
[23]imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.ADASYN.html
[24]imbalanced-learn.org/stable/references/over_sampling.html

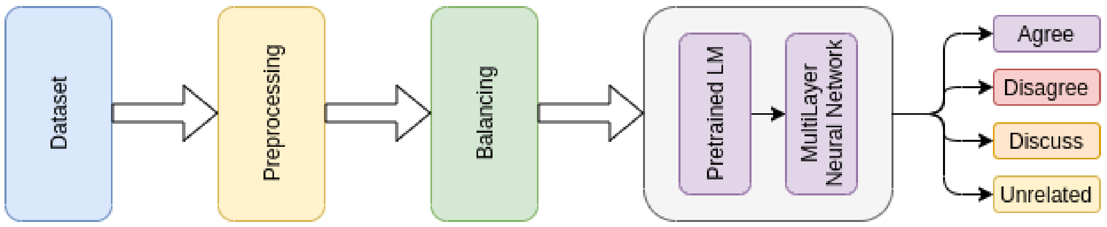


Figure 5.3: Schematic of each deep learning model.

5.6 Deep Learning

In deep learning approach, a combination of all predictors can be fed into the model and the model on its own will automatically learns which predictor is useful for the task. This property is the biggest advance of deep learning in comparison to machine learning (Deepak P [2021]). In contrary, in machine learning it was a critical step to design input predictors that the model can performs the best. And hours of trying different combination of predictors is needed (Giansiracusa [2021]). Schiller et al. [2020] assessed that, In contrast deep learning models, machine learning models that is trained on a single dataset, usually generalize poorly to other domains. The schematic of deep learning model is shown in figure 5.3. Headline-to-claim and Article-to-claim models have same schematic. Only some parameters vary in each model. In the following sections, pretrained language model used for stance detection are described.

5.6.1 BERT

ParsBERT (Farahani et al. [2020]) pre-trained model is used at the top of the model. ParsBERT model is a monolingual transfromer language model based on Google's BERT model Devlin et al. [2019]. Pretrained BERT model is adopt from HooshvareLab[25]. BERT is used to extract in context features from the corpus. Input of the model are *input ids*, *token type ids*, and *attention mask*. Then, two dense layer and another dense layer as the classifier is added to the output of the BERT model.

5.6.2 ALBERT

In this experiment, ALBERT (Lan et al. [2020]) model is substitute with BERT model in figure **??**. Other configurations are same as BERT experiment.

[25]huggingface.co

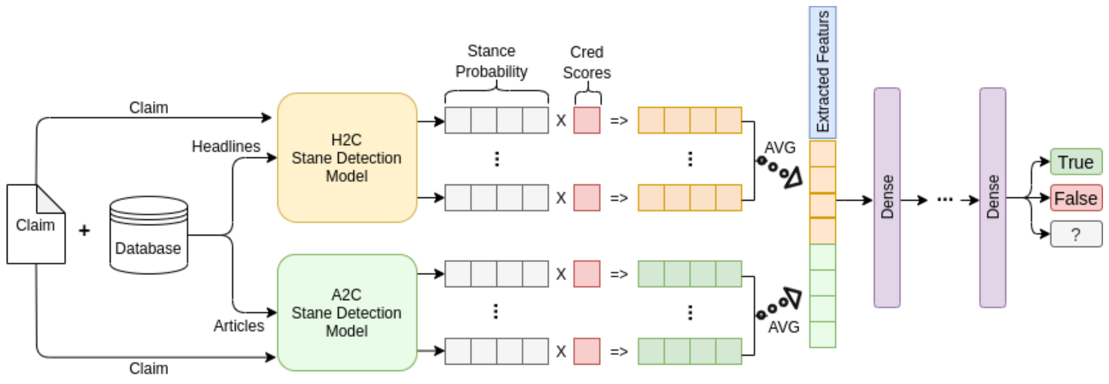


Figure 5.4: Schematic of each fake news detection model.

5.7 Article to Claim

Deep learning models perform far better on language inference tasks so they are better choices for article-to-claim stance classification. To calculate stance of a claim toward the body of a news, three different model based on a pre-trained Persian language model based on BERT, ParsBERT, and ALBERT are evaluated against each other.

5.8 Fake News Detection

Shematic of fake news pipeline is shown in figure 5.4. To detect a news article veracity, headline-to-claim and article-to-claim stance detection models, are considered as a black box. Four news articles is considered to evaluate veracity of a claim. Firstly, stance of a claim toward each headline of desired news articles and the body of the news article predict by stance classifier models. All predicted stance vectors are concatenated and along with some other features are fed into a multi-layer perception network in order to predict the claim veracity.

Credibility of news websites is one of the most important extracted feature. The credibility can be calculated through the following steps (The credibility score of head-claim and article-claim is calculated similarly except using their related stance ground truth):

- **Initialization**: The credibility score of all news websites that are existing in the Majid Zarharan [2019] dataset is set to zero at first. For the test set or predicting new samples, if the website doesn't exist in the data set, credit score is set to 0.1.

- **Quantification:** For each sample in the dataset, credibility score changes according to Table 5.1. ρ value is calculated from equation 5.11 if it is needed.

Table 5.1: Value of credibility according to GroundTruth and Veracity labels.

GroundTruth	Veracity	Value
Agree	True	$+1$
Disagree	False	$+1$
Agree	False	-1
Disagree	True	-1
Discuss	True	$+\rho$
Discuss	False	$-\rho$
Unrelated	-	No change
-	Discuss	No change

$$\rho = \frac{P(x, Agree) - P(x, Disagree)}{P(x, Agree) + P(x, Disagree)} \tag{5.11}$$

where:

$P(x, Agree)$	— Probability of Agree
$P(x, Disagree)$	— Probability of Disagree

- **Score Calculation:** The credibility score for each news website is calculated from equation 5.12.

$$H(X) = \frac{\sum_{i=1}^{k_X} credibility\, of\, x_i}{k_X} \tag{5.12}$$

where:

X	— A news website
k_x	— The number of news article in the dataset from x
$credibility\, of\, x_i$	— Due to table 5.1

Also another features are extracted to detect fake news, such as :

- One-hot encoding of the news website domain

- The ratio of samples that have been properly labeled as agree or disagree to the total sample of the news website (Correct ratio)
- The ratio of samples that have been wrongly labeled as agree or disagree to the total sample of the news website (Wrong ratio)
- The ratio of the total number of news website articles to the total number of articles in the data set.

Chapter 6

Results

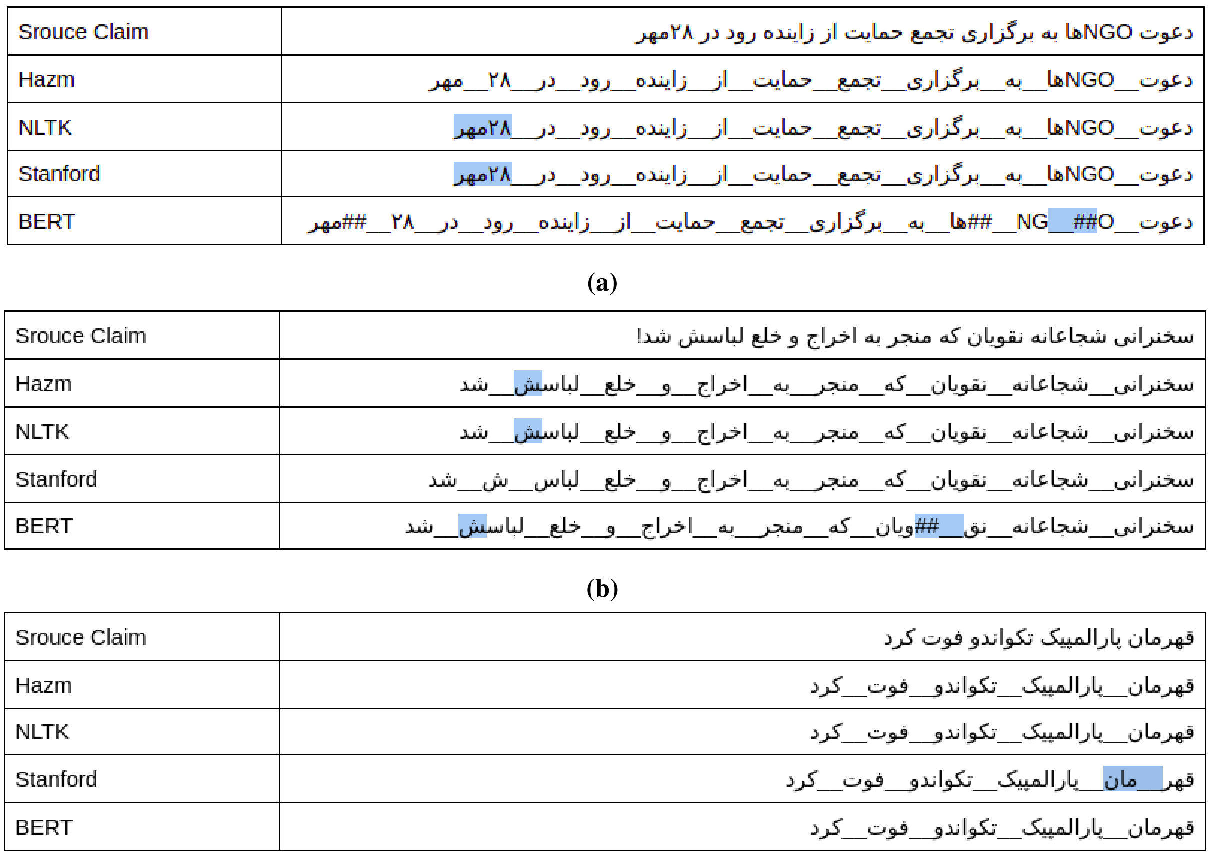
In this section, the results of the experiments of each step are discussed. Besides, different ideas and algorithms which are described in section 5 are evaluated. At first, machine-learning-based experiments are presented. As machine learning feature engineering, requires multiple experiments to find best working predictors, there are many trials and errors for a variety of algorithms and ideas. Then, Deep learning models. are compared to machine learning models.

6.1 Tokenizetion

Tokenization of *Hazm*, *Stanford*, *NLTK*, and *BERT* are evaluated against each other manually. Every single difference is evaluated[1]. Only three cases are presented as examples to compare each tokenizer's performance in figure 6.1. We need to remove particular words and patterns from the corpus so it is important to find a tokenizer that distinguishes all words correctly. Besides, it shouldn't lose information while tokenizing and be as fast as possible.

In figure 6.1, unsuitable tokens are highlighted blue. There isn't any flawless algorithm. As BERT is subwords, there are some UNK tokens and wrongly words broken by BERT tokenizing. For example figure 6.1, part (b) blue highlighted word is broken into two pieces wrongly. Strength of *Stanford* tokenizer, is that consider connected pronoun individually. But sometimes Stanford wrongly breaks an original word with a wrong assumption that the desired word has a connected pronoun. Figure 6.1, part (c) is a sample

[1]All different cases during corpus tokenization by those 4 algorithms are gathered in here



Srouce Claim	دعوت NGOها به برگزاری تجمع حمایت از زاینده رود در ۲۸مهر
Hazm	دعوت__NGOها__به__برگزاری__تجمع__حمایت__از__زاینده__رود__در__۲۸__مهر
NLTK	دعوت__NGOها__به__برگزاری__تجمع__حمایت__از__زاینده__رود__در__۲۸مهر
Stanford	دعوت__NGOها__به__برگزاری__تجمع__حمایت__از__زاینده__رود__در__۲۸مهر
BERT	دعوت__NG__##O__##ها__به__برگزاری__تجمع__حمایت__از__زاینده__رود__در__۲۸__##مهر

(a)

Srouce Claim	سخنرانی شجاعانه نقویان که منجر به اخراج و خلع لباسش شد!
Hazm	سخنرانی__شجاعانه__نقویان__که__منجر__به__اخراج__و__خلع__لباسش__شد
NLTK	سخنرانی__شجاعانه__نقویان__که__منجر__به__اخراج__و__خلع__لباسش__شد
Stanford	سخنرانی__شجاعانه__نقویان__که__منجر__به__اخراج__و__خلع__لباس__ش__شد
BERT	سخنرانی__شجاعانه__نق__##ویان__که__منجر__به__اخراج__و__خلع__لباسش__شد

(b)

Srouce Claim	قهرمان پارالمپیک تکواندو فوت کرد
Hazm	قهرمان__پارالمپیک__تکواندو__فوت__کرد
NLTK	قهرمان__پارالمپیک__تکواندو__فوت__کرد
Stanford	قهر__مان__پارالمپیک__تکواندو__فوت__کرد
BERT	قهرمان__پارالمپیک__تکواندو__فوت__کرد

(c)

Figure 6.1: Comparison performance of *Hazm*, *Stanford*, *NLTK* and *BERT* tokenizers.

on wrong separating pronoun and at part (b), pronouns are separated correctly. It can be seen in figure 6.1 *Hazm* performance is highly similar to *NLTK*. The only difference is that in contrast *NLTK*, Hazm separates numbers and punctuation in the corpus (Figure 6.1, part (a)).

Besides, the duration of tokenizing for each tokenizer is compared in figure 6.2. While *Hazm* is the fastest words tokenizer among evaluated algorithms, *Stanford* tokenizer last vastly longer. According to all pieces of evidence, *Hazm* tokenizer is the best tokenizer for this task.

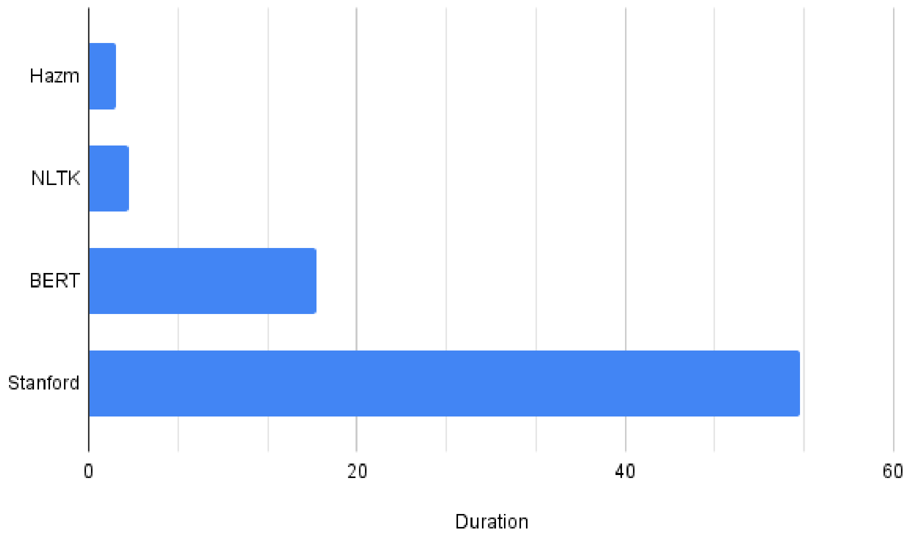


Figure 6.2: Comparison duration of tokenizing algorithm on Majid Zarharan [2019] dataset.

6.2 Stop-Words

Three sets of stop words is evaluated it this section. Majid Zarharan [2019] contains 1255 words which include wide a range of parts of speech. In contrast, NonVerbal[2] stop words set include 158 words and it doesn't support any verb. In the new version of stop-words, 18 stop words are removed from the Nonverval set. This set is called *shortened* and 82 new words are added. Finally, the *Extended* version contains 233 words. To evaluate the performance of each stop-words sets, all desired features with Tf-iDF as word representation after removing particular stop words are fed into an SVM (Chang and Lin [2011]) model. To having a fair comparison, non-targeting removing the stop-words is also compared.

It can be inferred from figure 6.3 that the list of stop words which is used by Majid Zarharan [2019] is ignoring valuable data and it is even better not to remove stop words. Nonverbal stop-words achieve higher accuracy on stance detection. Through, whether removing or adding words from Nonverbal (*shortened* version) didn't improve results. Altogether, Nonverbal list of stop-words, performs the best in this task.

[2]Gathered by Kharazi

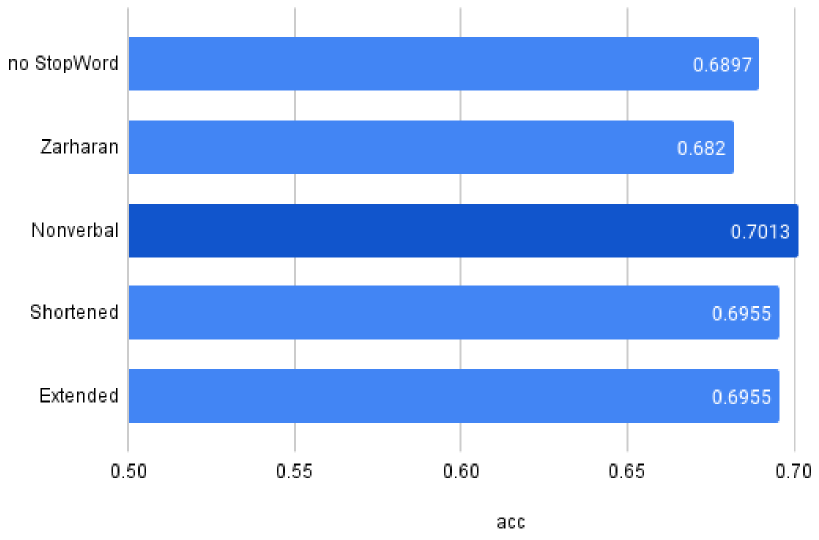


Figure 6.3: Comparison accuracy of SVM model with different configuration of stop words.

6.3 Word Representation

In this section, Bow (Harris [1954]), TF-iDF, and Word2Vec (Tomas Mikolov [2013]) performances are compared by SVM machine learning algorithm in the stance detection task. According to figure 6.4 Tf-iDF performs the best in comparison to BoW and Word2Vec in order to represent words.

6.4 Predictors

Different combinations of calculated predictor are run to find out that which set of predictors performs better. Desired predictors are Similarity, RootDistance, IsQuestion, HasTwoParts and Polarity. In this Section, both SVM (Chang and Lin [2011]) and RandomForest (Liaw and Wiener [2002]) results are considered in evaluation to have a more accurate analysis.

Firstly, both models are trained on the corpus representation by TF-iDF without any other predictor. Then, each predictor is added into TF-iDF vector to evaluate their effectiveness individually, and finally, all features together are fed to models. According to table 6.1, Similarity and ImportantWords have the highest positive effect respectively, on both accuracy (5.9) and f1-score (5.10). Similarity score has improved accuracy 15 to 17 percent and ImportantWords has improved SVM accuracy by almost 4 percent. Though,

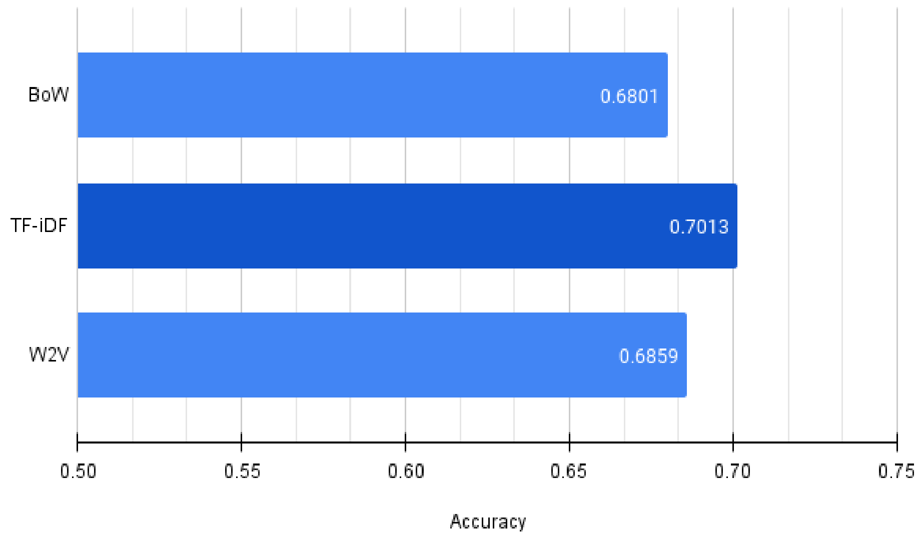


Figure 6.4: Comparison SVM model with Bow, TF-iDF and Word2Vec word representaion algorithms.

predictors such as IsQuestion, HasTwoParts, and Polarity don't have a significant effect on results, using them all together boost total accuracy and f1-score.

Due to figure 6.1, using RootDistance, IsQuestion, and HasTwoParts individually decreases accuracy, so models are also trained with two other variations. Though IsQuestion, HasTwoParts, and polarity don't have a positive effect individually, using them with Similarity, ImportantWrods, and polarity boost accuracy. On the other hand, removing RootDistance from All predictors mode whether have a negligible effect or improves accuracy.

According to previous comparisons and evaluation inferred from table 6.1, the best predictors to use for stance detection task is the combination of Similarity, ImportantWords, IsQuestion, HasTwoParts, and Polarity predictors in addition to TD-iDF as the corpus representer, altogether.

6.5 Machine Learning

In this section, each Gaussian Naive Bayes, SVM, Linear SVC, Random Forest, and Logistic Regression parameters are tuned on stance detection task with respect to chosen predictors in the previous section (6.4). In the next step, the performance of models is compared to each other.

Table 6.1: Comparison of accuracy and F1-score with different combinations of predictors for both SVM and Random Forest classifiers.

Predictors	SVM		RandomForest	
Model	Acc.	F1.	Acc.	F1.
TF-iDF only	51.83	51.90	52.79	54.00
+ Similarity	66.85	66.71	68.78	67.88
+ Root Distance	51.25	51.50	49.71	50.52
+ Important Words	56.64	56.94	52.98	52.49
+ Is Question	51.63	51.79	51.63	52.70
+ Has Two Parts	51.83	51.90	50.28	50.87
+ Polarity	52.21	52.50	52.40	53.25
+ All	69.74	69.75	69.36	68.75
+ All - Root Distance	69.74	69.69	**70.71**	**70.28**
+ Similarity + ImportantWords	69.74	69.69	67.82	67.02

6.5.1 SVM

Three SVM classifier tuned parameters are Kernel, Regularization parameter (C) and degree of polynomial kernel. Evaluated kernels are RBF, Polynomial and Sigmoid. According to figure 6.5, Sigmoid works weak for this task. Each polynomial kernel behaves differently due value of the regularization parameter. Polynomial with degrees 2 and 3 performs better than 1 and 4. Linear polynomial may be so simple and SVM is not good enough to classify stance with polynomial with degree 4. RBF, Polynomial degree 3 behave similarly due to the regularization parameter changes.

The best configuration for SVM classifier is using RBF as the kernel with regularization parameter equal to 2.5 confirming figure 6.5. Learning procedure and details of each model training exist in this project GitHub repository[3].

6.5.2 Linear SVC

Linear Support Vector Machine Classifier algorithm, loss function and Regularization parameters are tuned with penalty equals to *l2*. Figure 6.6 illustrates comparison between linear svm models with loss functions

[3]Different SVM configuration training details [here].

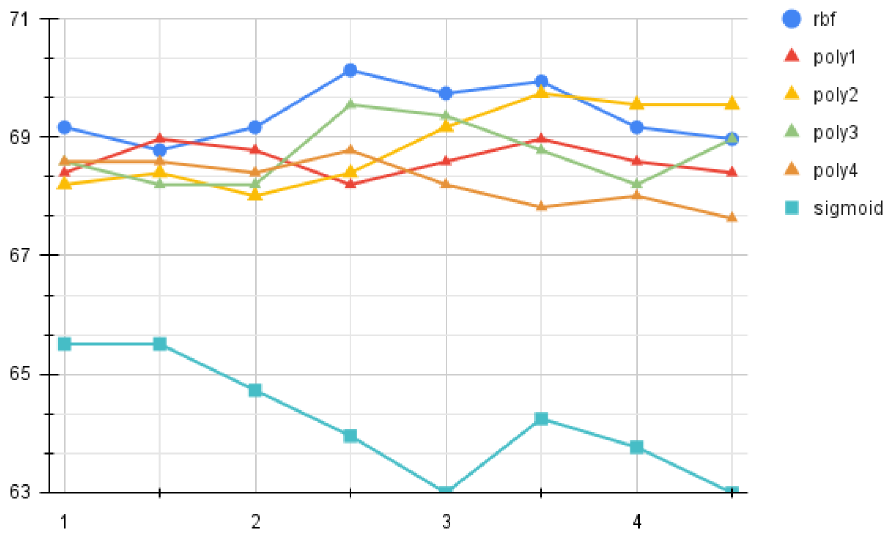


Figure 6.5: Tuning SVM model parameters with TF-iDF representaion algorithms

equal to Hinge or Squared Hinge, and tuned SVM algorithm. Hinge loss function has scored better performance than Squared Hinge. When Squared Hing is used as loss function, as the Regularization parameter increases, accuracy score decrease. Though Regularization parameter don't have significant effect on accuracy score when loss is equal to Hinge. In conclusion, due to figure 6.6 Best configuration is for Hinge loss and Regularization parameter equals to 1.0.

6.5.3 Random Forest

Random Forest criterion, maximum number of feature in each decision tree and number of tree are evaluated. Figure 6.7 illustrates effect of number of tree in forest on accuracy among different configuration. Accuracy of models increase on average by adding more trees into the forest. Besides, *gini* algorithm performs better than *entropy* to measure quality of splits. Also, three different upper bound is considered for number of features when looking for a split. No boundary, *sqrt* of total feature and *log2* of total feature. According to figure 6.7 don't applying any boundary leads to better accuracy in average. Furthermore, as the boundary get tighten average performance decreases.

Best configuration for Random Forest machine learning model in this task is using *gini* algorithm to evaluate splitting quality, not applying any boundary on number of features and having 125 decision tree in

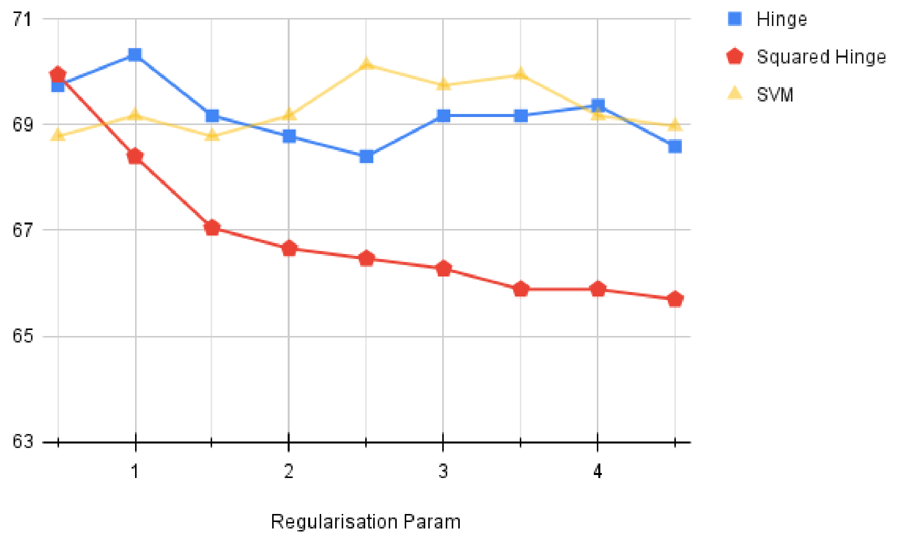


Figure 6.6: Tuning LinearSVC model parameters with TF-iDF representaion algorithms.

the forest.

6.5.4 Logistic Regression

Many experiments are designed to evaluate behavior of Logistic Regression models. *Elasticnet* (Equation 5.8) penalty algorithm which is used in penalization procedure, used for *saga* solver and *l2* penalty algorithm is used for *sag*, *lbfgs*, and *newton-cg* solvers.

Elasticnet has a ρ parameter which determines portion of using $l1$ to $l2$ penalty in *saga* solver. Figure 6.8 illustrate effect of ρ values from 0 to 0.9 on accuracy of stance detection. Besides, models with regression parameter from 0.5 to 4.5 are evaluated from determined ρ range. It can be inferred from figure 6.8 that ρ parameter doesn't have significant effect on accuracy of the model. While, regression parameter between 1 and 2.5 clearly results in higher accuracy rather than external range. It can be also inferred from figure 6.9 which illustrates effect of regression parameter on stance classification accuracy. Models with different value of ρ behave similarly and best performances happens when regression parameter is between 1 and 2.5.

Another variants of Logistic Regression setup is to use $l2$ penalty with desired solver algorithms. Figure 6.10 compares best *saga* solver with *sag*, *lbfgs*, and *newton-cg*. The Logistic Regression with *lbfgs* solver

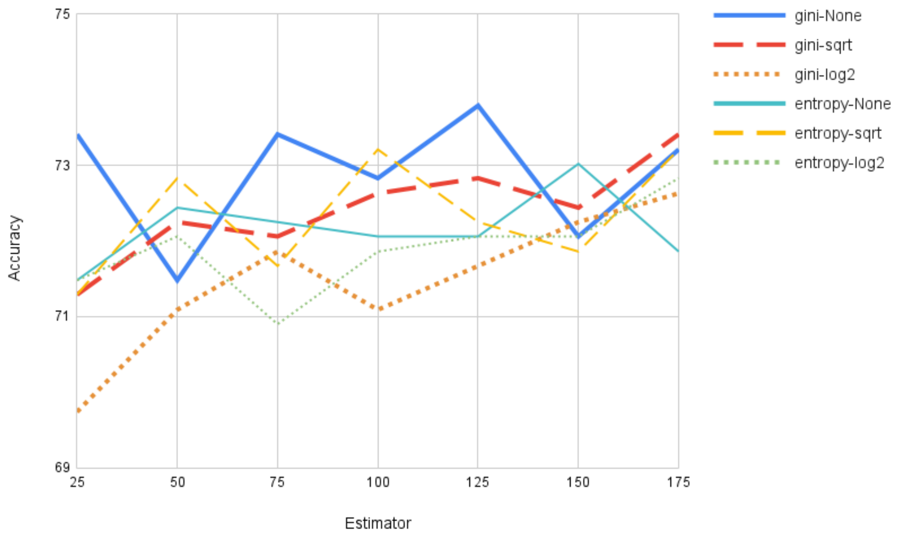


Figure 6.7: Random Forest machine learning model different configuration on stance detection task. Type of line presents type of boundary applied on each model. Solid line, dash line and doted line stands for no boundary, sqrt of total feature and log2 of total feature respectively.

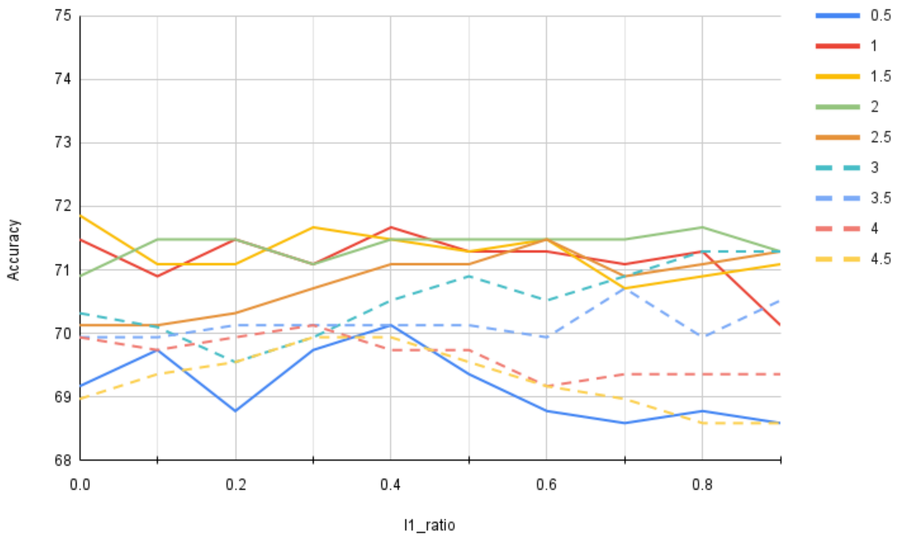


Figure 6.8: Effect of ρ parameter of *elasticnet* penalty on stance detection task.

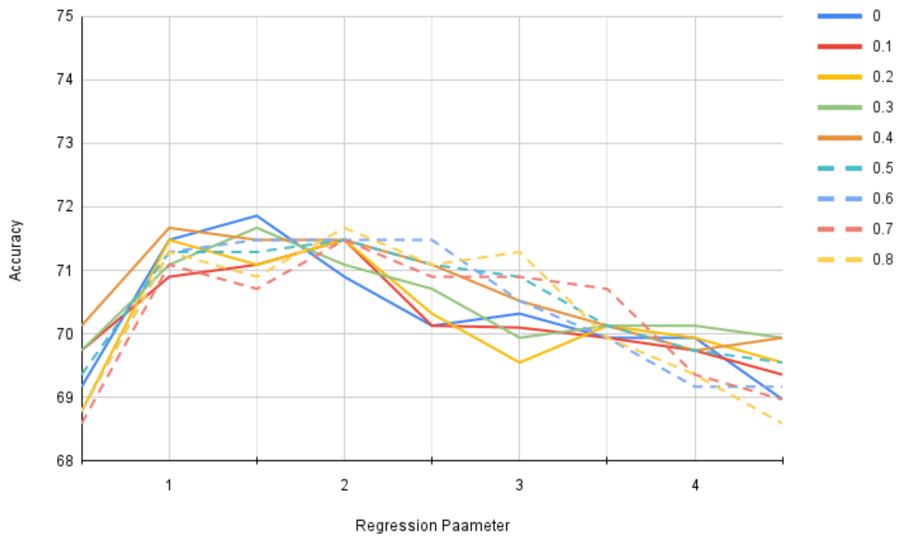


Figure 6.9: Effect of regression parameter of *elasticnet* penalty on stance detection task.

and regression parameter equals to 1.5 has recorded the highest accuracy.

6.5.5 Comparison

In previous sections parameters of each SVM, LinearSVC, Random Forest, and Logistic Regression models are tuned. In this section models with desired parameters has run 5 times each to have more reliable results and comparison. Average accuracy and highest accuracy recorded in tuning phase, compared in figure 6.11. The highest achievable accuracy with machine learning models to classify stance of a claim towards the headline of a news article is 74.01%.

6.6 Dataset Balancing

Firstly, ADASYN, SMOTE, SVMSMOTE, BorderLineSmote, and RandomOverSampler oversampling methods are applied on the Majid Zarharan [2019] dataset. Each method is evaluated against five desired machine learning models. Red sereis in figure 6.12 stands for the accuracy of models, associated with the Majid Zarharan [2019] dataset. LinearSVC, LogisticRegresion, and GaussianNaibeBayes are not compatible with any oversampling method. Though, ADASYN oversampling method has increased these two model accuracy 5 percent on average.

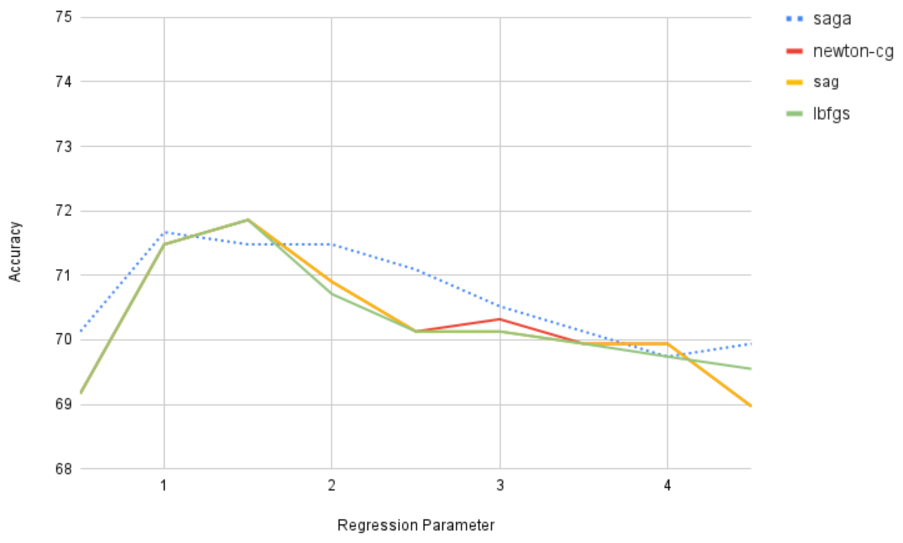


Figure 6.10

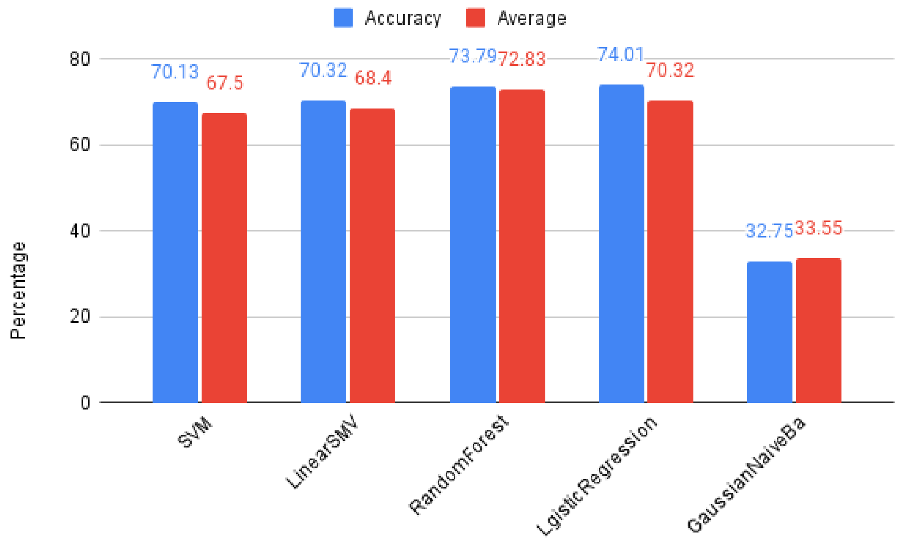


Figure 6.11: Comparison SVM model with Bow, TF-iDF and Word2Vec word representation algorithms.

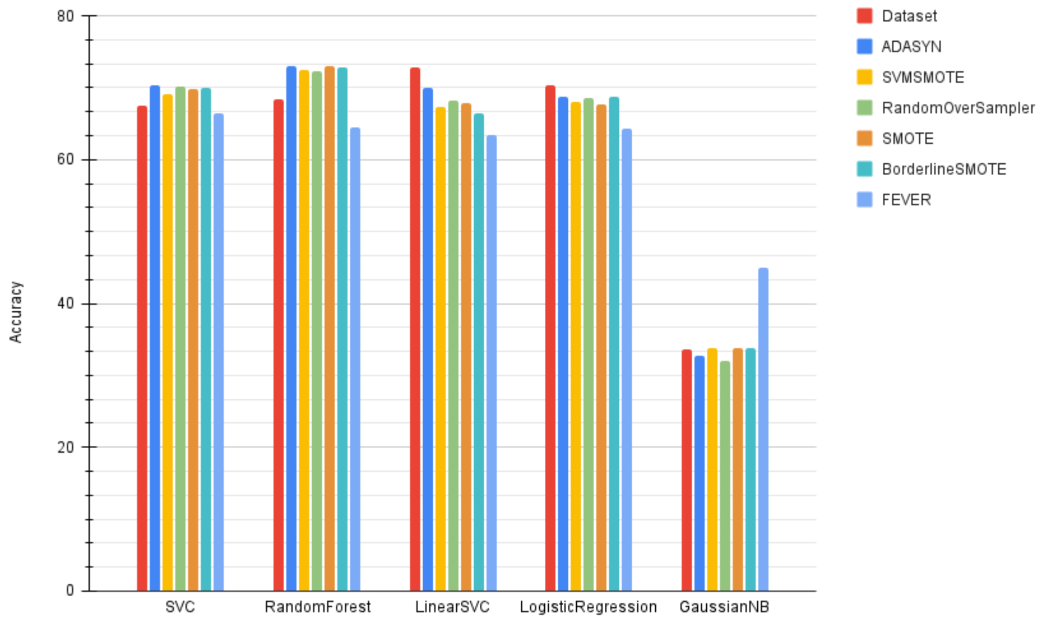


Figure 6.12: Comparison SVM model with Bow, TF-iDF and Word2Vec word representation algorithms.

At the second step, dataset is extend by the ParsFever (Zarharan et al. [2021]) dataset. Though, number of samples are increased accuracy is obviously decreased, but the GaussianNB model. This may happens that data sources from each dataset are totally different and headlines in ParsFever are much longer than the Majid Zarharan [2019] dataset.

6.7 Deep Learning

Pre-trained BERT model is used at the top of the model, then two Dense layers and finally a Dense including 4 neuron to classify stance is considered for end-to-end system. Each epochs lasts about 26 seconds in training procedure. Figure 6.13 illustrates the model training procedure. In comparison to machine learning models, the deep learning model has boosted the accuracy of head to claim stance detection by 10 percent.

BERT-based model has learned for 20 epochs. Validation loss has stated to increase since epoch 11 and validation accuracy hasn't changed considerably then. Best validation accuracy has converged on 80.92% on the headline-to-claim dataset.

ParsBert-based model has learned for 20 epochs. Best validation accuracy has converged on 81.11% on

Table 6.2: comparison of headline-to-claim stance detection models.

Model	F1	Accuracy	F1	Accuracy
SVM+ADASYN	69.63	69.15	69.38	70.32
RandomForest+ADASYN	71.24	69.14	70.17	73.02
BERT	81.65	80.69	81.16	80.92
ParsBERT	84.67	79.42	81.96	81.11
ALBERT	75.75	64.09	69.43	70.52
ParsBERT+ADASYN	84.96	85.64	85.29	**85.48**

Table 6.3: Comparison of article-to-claim machine learning and deep learning models.

	Headline to claim		Article to claim	
Model	Precision	Recall	F1	Accuracy
SVM+ADASYN	69.63	69.15	69.38	70.32
ParsBERT	69.63	69.15	69.38	70.32
ParsBERT+ADASYN	69.63	69.15	69.38	70.32

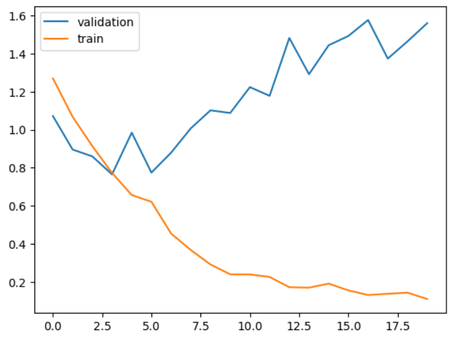
the headline-to-claim dataset. In comparison to machine learning algorithm, deep learning algorithm has enhanced about 10% accuracy. Besides, loss score has converged on a lower score than BERT-based model.

Though, best recorded ALBERT language model on 20 epochs is at most 70.52% on accuracy score. Figure 6.13, part (e) and (f) is illustrated training procedure.

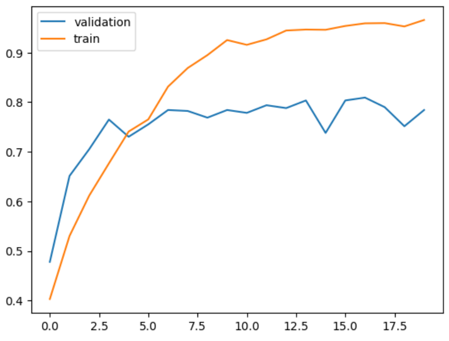
Among these three alternative of BERT algorithm, PasrBERT based model has recorded best accuracy score with 85.48% accuracy on stance prediction.

6.8 Article to Claim

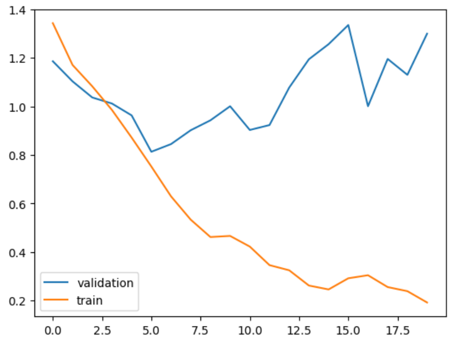
Best stance detection models on both machine learning and deep learning models are evaluated on article-to-claim task. Table 6.3 show those models performance on both headline-to-claim and article-to-claim task.



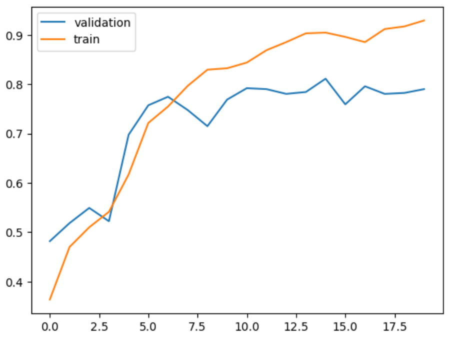
(a)



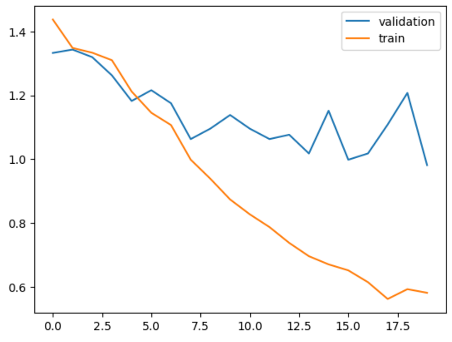
(b)



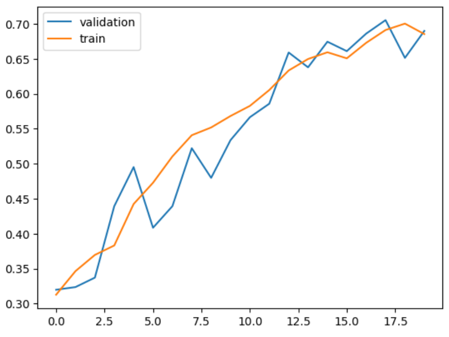
(c)



(d)

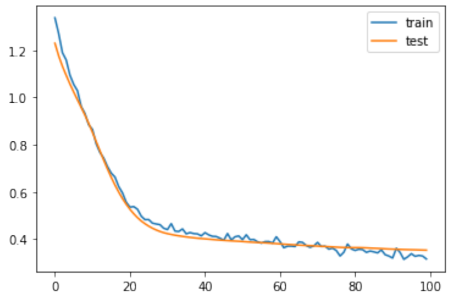


(e)

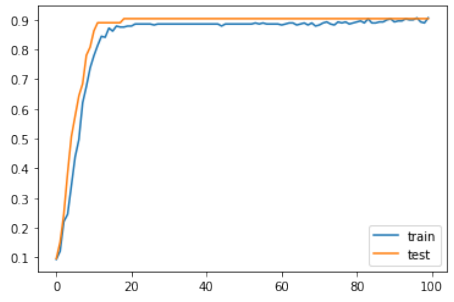


(f)

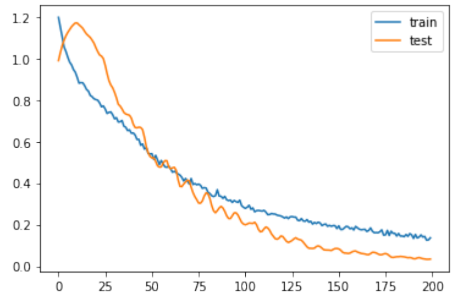
Figure 6.13: Deep learning procedure on the headline-to-claim stance detection task. Left figures illustrate loss score and right figures illustrate accuracy score of train and test data during training procedure. (a, b) Pre-trained language model based on Google's BERT (Farahani et al. [2020]) on Persian corpus (c, d) Pre-trained monolingual language model based on ParsBERT (Farahani et al. [2020]) on Persian corpus. (e, f) Pre-trained language model based on ALBERT (Lan et al. [2020]) on Persian corpus



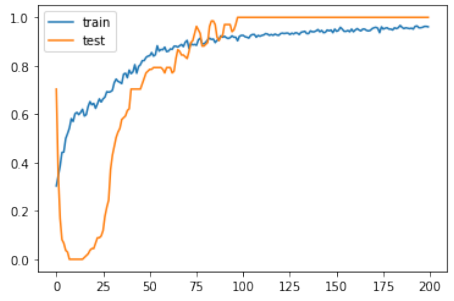
(a)



(b)



(c)



(d)

Figure 6.14: Left figures illustrate loss score and right figures illustrate accuracy score of train and test data during training procedure for each iteration. (a, b) Training procdure on fake news detection model trained on the Majid Zarharan [2019] dataset. (c, d) Training procdure on fake news detection model trained on oversampled Majid Zarharan [2019] dataset by ADASYN (He et al. [2008]) algorithm.

6.9 Fake News

the best BERT-based model for headline-to-claim and article-to-claims are considered for this part. These model prediction base on 4 news articles are concatenated with features which are described in section 5.8. Then the overall vector is feed to an three-layer MLP model as the classifier. ADASYN oversampling method is also used to deal with imbalanced classes. Figure 6.14 illustrates training procedure before oversampling and after oversampling the dataset. After oversampling, the trained model accuracy has converged to 99%, while before oversampling model stops at 90.41%. According to figure 6.14, after adding oversampled samples, at first it is harder for model to decease the loss score on the train set and loss score convergent takes longer time. Though after balancing the datast (Figure 6.14, part (a)), the value of loss score has converged 0.2 lower than the origin dataset (Figure 6.14, part (c)).

Chapter 7

Conclusion

Machine learning models are evaluated on Persian stance detection task as baseline. Multiple predictors are extracted and different combination of them are applied on machine learning models to find out the most effective predictor combination. Due imbalanced samples distribution in the Majid Zarharan [2019] dataset, extending the dataset by ParsBERT dataset (Farahani et al. [2020]) and oversampling methods are discussed and compared to each other.

In the Persian stance detection task, using deep language models boosts performance noticeably. BERT and ParsBERT as it's alternative have high ability to make inference from a given content so they can represent current content sufficiently and test accuracy in the model based on ParsBERT language model is equal to 85.48%. As the result, BERT language models predict stance detection 10% higher than machine learning algorithm on average. In contrast, requires time consuming feature engineering and parameter tuning and they can't achieve high accuracy on language inference tasks. For both machine learning and deep learning models performance decrease on article-to-claim task. Despite of this issue, machine learning algorithm have achieve even higher than 70% accuracy score on headline-to-claim task. 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]).

The best pretrained stance detection model on headline-to-claim and article-to-claim are separately utilized in fake news detection. We have achieve 99% accuracy score on the Majid Zarharan [2019] dataset. Though, number of samples in the dataset gathered by Majid Zarharan [2019] contains 1624 samples. Increasing number of samples help model to improves model generalization.

Adding attention layers to the classifier have improve results in current researches. Fox instance, Shu et al. [2020] alleged that using attention layers in the model have make a great contribution on accuracy scores. Utilizing such models can also make increase on the ability of these models to go beyond the current state in such low-resource settings. Besides, increasing number of samples in the Majid Zarharan [2019] dataset toward achieving balanced dataset can help the model to distinguish each class precisely.

References

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.

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

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.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. *ArXiv*.

Andy Liaw and Matthew Wiener. 2002. Classification and regression by randomforest. *R News*, 2:18–22.

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.

Kai Shu, Suhang Wang, Dongwon Lee, and Huan Liu. 2020. Disinformation, misinformation, and fake news in social media.

Shivangi Singhal, Rajiv Ratn Shah, Tanmoy Chakraborty, Ponnurangam Kumaraguru, and Shin'ichi Satoh. 2019. Spotfake: A multi-modal framework for fake news detection. In *2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM)*, pages 39–47.

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.

Iran University of Science and Technology

Computer Engineering Department