Persian Fake News Detection using Stance Classification

Mahsa Ghaderan

A Thesis

submitted in partial fulfillment of the

requirements for the degree of

Bachelor

Iran University of Science and Technology
2021

Reading Committee:
Main Guardian, Chair
Dr. Sauleh Eetemadi
Dr. Behrouz Minaei-Bidgoli

Program Authorized to Offer Degree:
Computer Engineering

Iran University of Science and Technology

Abstract

Persian Fake News Detection using Stance Classification

Mahsa Ghaderan

Chair of the Supervisory Committee:

Dr. Sauleh Eetemadi Main Guardian
School of Computer Engineering

These days increase in unauthenticated internet sources has led to spreading fake news vastly. Failure to detect misinformation promptly can have signicantly irreparable damages on the walk of life. Recent researches have improved stance classication as a primitive step to detect fake news in English. Moreover, they have less focus on recognizing fake news. In this work, we have developed a deep learning model based on ParsBERT to improve the accuracy of the Persian stance classication. We over-sampled the available imbalanced dataset in Persian to compensate lack of data in such a low-resource language. Then, we implemented a model to detect fake news in Persian by using our best stance classifier and other manually extracted features. Consequently, we achieved an accuracy of 85% for stance classification and 99% for fake news detection on the employed dataset.

Keywords: Machine Learning, Deep Learning, Natural Language Processing, BERT, Oversampling, Stance Classification, Fake News Detection

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 this report.

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

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

I would like to dedicate this project to my lovely parents. You stand beside me in every single moment of my life. I wish this achievement completes a part of beautiful wishes you made for me.

Contents

List of Abbreviations

A2C Article to Claim

Acc. Accuracy

ADASYN Adaptive Synthetic Sampling

ALBERT A Lite BERT

BERT Bidirectional Encoder Representations from Transformers

BoW Bag of Words

DL Deep Learning

FNC-1 Fake News Challenge stage 1

H2C Headline to Claim

MDL Multi Dataset Learning

ML Machine Learning

NLTK Natural Language Toolkit

ParsBERT Persian Bidirectional Encoder Representations from Transformers

RBF Radial Basis Function

SAG Stochastic Average Gradient descent

SAGA A variant of SAG

SVC Support Vector Classification

SVM Support Vector Machine

TF-IDF Term FrequencyInverse Document Frequency

UNK Unknown

W2V Word2Vec

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 or decisions to influencing momentous events such as political elections (Mohtarami et al. [2018]). As days go by, the importance of automatically detecting stance is attracting 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 classify the relationship between a claim and a given text. Four considered labels are *Agree*, *Disagreed*, *Discussed*, and *Not Enough Information*. So it is possible to evaluate the orientation of a news source towards a particular issue Riedel et al. [2017]. The claim could be a sentence, 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, or any available information. Choosing the source and context of the sentence and the text depends on the goal of the defined task.

Gathering sufficient amount of data is a vital step to achieve a reliable predictor model. Both numbers of records in the dataset and the quality of each sample have a noticeable effect on stance prediction accuracy. Dulhanty et al. [2019]) gathered fifty thousand article-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, there is not enough available data in some contexts. Here is where transfer learning methods play a key role and compensate lack of data. Schiller et al. [2020] improved stance

detection model accuracy by fine-tuning BERT model. Also, Dulhanty et al. [2019] used RoBERTa model which is pre-trained on Facebook data. Then fine-tune it on the specified current task. In this project, we adopted a pre-trained model on a large text corpus in a general context and then fine-tune the model on task-specified data.

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 using recurrent neural networks such as LSTM and BiLSTM (Mrowca et al. [2017]), BERT, and utilizing attention-based networks (Mrowca et al. [2017]). Besides, Mohtarami et al. [2018] currently proposed a novel model architecture based on memory networks.

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

In this word, we have used Zarharan et al. [2019] dataset. This dataset has mainly focused on Persian news and rumors. At first, we have trained a stance classifier model for a given claim toward a headline or claim of a news article. Machine learning and deep learning approaches are evaluated. Besides, we have dealt with the imbalanced dataset by extending and oversampling the dataset. Then these two models take part in our fake news detection model.

In chapter 2 concepts and tools which are utilized in this project are described briefly. Then currently researches based on the stance classification and the fake news detection tasks are studied in chapter 3. In chapter 4 methods and ideas applied in this project are described in detail. After that chapter 5 contain experiments and their results individually and in comparison to each other. We have discussed those results in this chapter. Finally, there are an overview of what we have done and achievements in chapter 6.

Chapter 2

Concepts and tools

A brief description of adopted algorithms which are used in this project are available in this chapter. Algorithms are classified as *Tokenization*, *Words Representaion*, *Machine Learning*, *Oversampling*, *Language Models*, and *Evaluation Metrics*.

2.1 Tokenization

It is vital in the tokenizing step to break the corpus into correct words. Tokenizers limit the number of words to reduce computational costs, so it may happen that tokenizers generate meaningless words if the number of accepted words has not been set sufficiently. Also, sometimes tokenizers have not seen words before, and omit them while tokenizing the corpus. According to stated reasons, it is important to choose a tokenizer carefully.

2.1.1 NLTK

NLTK[1] (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. Natural Language Toolkit (NLTK) supports various tokenizer methods. Its basic algorithm tokenize words depends on white spaces.

[1] nltk.org

2.1.2 Hazm

Hazm[2] 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[2].

2.1.3 Stanza

Stanford NLP group has released Stanza tool[3]. Stanza is also another advantageous NLP tools package and can diagnose multi-word tokens. At first, it splits raw input text into sentences, then performs sentences segmentation. It is efficient for linguistic analysis and supports more than 70 human languages and releases new versions constantly.

2.1.4 WordPiece

WordPiece tokenizer used by BERT (Devlin et al. [2019]) machine learning model. BERT is a transformer-based machine-learning model which is currently used in various natural language processing tasks. WordPiece algorithm starts learning tokenization with a list of available characters in the text. Then it expands its word list by merging the most repeated group of characters. This procedure continues until hitting the considered number of words limit.

2.2 Word Representation

To represent a corpus, each token should be converted to a number or vector. Good representation carries semantic of each word or n-grams, sequential words contents, and is as brief as possible. Three baseline Bag-of-word (Harris [1954]), TF-IDF (Sammut et al. [2010]), and W2V (Mikolov et al. [2013]) algorithms are used in this project. In this section, these algorithms are described briefly.

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

2.2.1 Bag of Word

Bag-of-Words (Harris [1954]) model, is a way to represent texts. BoW keeps words frequency and dismisses word's orders and context. 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 vector 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[4].

2.2.2 Term FrequencyInverse Document Frequency

Term FrequencyInverse Document Frequency (Sammut et al. [2010]) is an algorithm to present the importance of each word in a corpus in a statistical way (TF-IDF). 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 2.1

$$\text{tf}\,(t, D) = \frac{f_{t,d}}{\Sigma_{t' \in d} f_{t',d}} \tag{2.1}$$

- **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 2.2.

$$\text{idf}\,(t, D) = \log \frac{N}{|\{d \in D, t \in d\}|} \tag{2.2}$$

After calculating *TF* and *iDF*, TF-IDF value for each word calculates from equation 2.3.

$$\text{tf-idf}\,(t, d, D) = \text{tf}\,(t, D)\,.\text{idf}\,(t, D) \tag{2.3}$$

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

2.2.3 Word2Vec

Word2vec (Mikolov et al. [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. W2V 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 W2V exist. For example, Fasttext is an extension of W2V model which treats each word as concatenated n-grams[5].

2.3 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. In this section, a brief overview of *Gaussian Naive Bayes*, *SVC*, *LinearSVC*, *Random Forest*, and *Logistic Regression* is available

2.3.1 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[6]:

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

[5] en.wikipedia.org/wiki/Word2vec
[6] en.wikipedia.org/wiki/Naive_Bayes_classifier

where:

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

y — Label of a class.

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

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

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

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

2.3.2 Support Vector Machines

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[7].

Regularization parameter stands for strength of regularization. Kernel of SVM specifies the type of separator that the SVM algorithm uses to distinguish different classes. It is a basic form of the mathematical function that SVM tunes during the training procedure. Kernels of SVC could vary from a simple polynomial such as *linear* to more complex ones such as *Radial basis function*, *Sigmoid*, and *Higher degree polynomial*. Furthermore, the wight of each class could be different from each other. *Balanced* mode set different weight for each class during training to compensate imbalanced data.

[7] wikipedia.org/Support-vector_machine

2.3.3 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 its kernel is equal to linear hyperplane polynomial.

2.3.4 Random Forest

Random Forest (Ho [1995]) 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.

2.3.5 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.

Logistic Regression has a number of parameters can be tuned due to the specified task. Penalization and regularization parameter could be *l1* (Equation 2.4), *l2* (Equation 2.5), or *elasticnet* (Equation 2.6). If ρ parameter equals to 0.5 in Elastic-Net Penalization, means $l1$ and $l2$ have the same powers.

- $l1$

$$min_{\omega,c}\ \frac{1}{2}\omega^T\omega + C\Sigma_{i=1}^{n}\log\left(\exp\left(-y_i\left(X_i^T\omega + c\right)\right) + 1\right) \tag{2.4}$$

- $l2$

$$min_{\omega,c}\ ||\omega||_1 + C\Sigma_{i=1}^{n}\log\left(\exp\left(-y_i\left(X_i^T\omega + c\right)\right) + 1\right) \tag{2.5}$$

- $elastic - net$

$$min_{\omega,c}\ \frac{1-\rho}{2}\omega^T\omega + \rho\,||\omega||_1 + C\Sigma_{i=1}^{n}\log\left(\exp\left(-y_i\left(X_i^T\omega + c\right)\right) + 1\right) \tag{2.6}$$

where:

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

y_i — Takes value between -1, 1.

Another parameter specifies optimizer algorithm could be Minimizing Finite Sums with the Stochastic Average Gradient (Stochastic Average Gradient descent (SAG)), A variant of SAG (SAGA) which is an alternative of SAG that supports $l1$ optimizer, Newtons method which utilize quadratic approximation, or Limited-memory BroydenFletcherGoldfarbShanno which is analog to Newton's method and Hessian matrix is approximated by updating gradient evaluation and it is more robust in larger datasets. Although, Limited-memory BroydenFletcherGoldfarbShanno is slower than *SAGA*[8].

2.4 Oversampling

A 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 datasets that suffer from lack of data. It is important to notice that splitting test and train sets should be performed before resampling, and oversampling should be only applied on the train set. RandomOverSampler, SMOTE, SVMSMOTE, BorderlineSMOTE, and ADASYN algorithms are described in the following sections.

2.4.1 Random Oversampler

This method randomly peak samples from classes and re-sample 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[9].

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

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

2.4.2 SMOTE

Synthetic Minority Over-sampling Technique (Chawla et al.) is an oversampling method by generating new samples by interpolation. It's not important for smothe sampler that point is chosen to be resampled.

2.4.3 SVM-SMOTE

SVMSMOTE (Demidova and Klyueva [2017]) is a variant of SMOTE oversampling method which uses SVM (Section 2.3.2) algorithm to choose resampling samples. One strength of this model is that it is effective for both vector data and sequence data (Demidova and Klyueva [2017])[10].

2.4.4 BorderlineSMOTE

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.[11]

2.4.5 Adaptive Synthetic Sampling

Adaptive Synthetic Sampling Approach for Imbalanced Learning (ADASYN) (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]).

2.5 Language Representation Model

Language representation models can represent a word embedding in context. Pretrained language models have significant improve on many natural language model tasks such as natural language inference task (Devlin et al. [2019]). BERT, ParsBERT, and ALBERT models are described in short in the following sections.

[10] imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SVMSMOTE.html
[11] imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.BorderlineSMOTE.html

2.5.1 BERT

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 (P et al. [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 text both left to right and right to left.

2.5.2 ParsBERT

ParsBERT (Farahani et al. [2020]) pre-trained model for Persian language that can be 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] which is lighert than multilingual BERT. Pretrained ParsBERT model is available on HooshvareLab[12]. ParsBERT outperforms other language model variants on Persian.

2.5.3 A Lite BERT

A Lite BERT (Lan et al. [2020]) model is a lighter variant of BERT model. The large BERT model has 9 times more parameters than ALBERT. ALBERT is released by Google's team in 2019. It separates input and hidden layers embedding. The input layer is context-independent while the hidden layer is context-dependent. Same as BERT, ALBERT uses masked language model but instead of NSP (Next Sentence Prediction) ALBERT uses SOP (Sentence Order Prediction) which only looks for sentence coherence in contrast to SOP which looks for both coherence and the topic to identify the next sentence.

[12]huggingface.co

2.6 Evaluation Metrics

The accuracy score is calculated from 2.7. It is good to evaluate the ratio of correctly predicted samples. Though accuracy score is not reliable to evaluate model performance specially when dealing with an imbalanced dataset. In addition to correctly predicted samples, False model prediction has an impact on the final F1 score (2.8).

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{TN} + \text{FP}} \tag{2.7}$$

where:

TP — True Positive

TN — True Negative

FP — False Positive

FN — False Negative

$$\text{F1} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precission} + \text{recall}} \tag{2.8}$$

where:

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

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

As the result, F1-score is a balanced measure of the model performance. F1-score reveals negative aspect performance of the model.

Chapter 3

Related Work

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 Fake News Challenge stage 1 (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.

[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] Pedregosa et al. [2011]

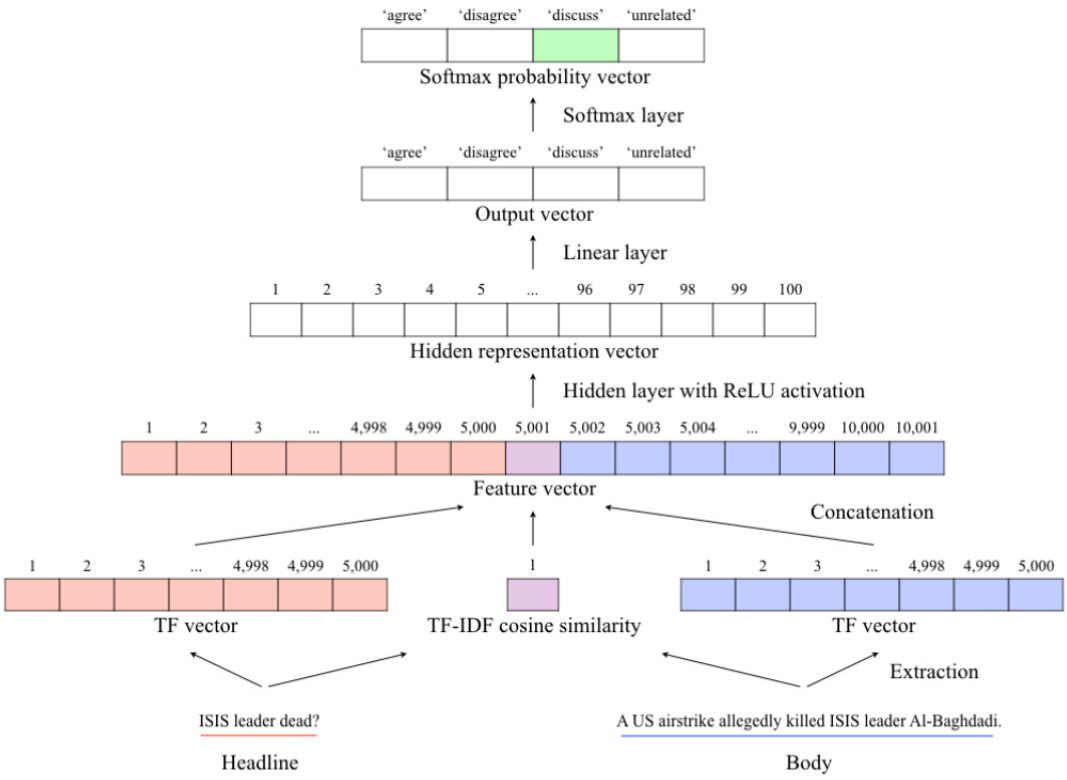


Figure 3.1: Schematic diagram of Riedel et al. [2017] model.

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 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

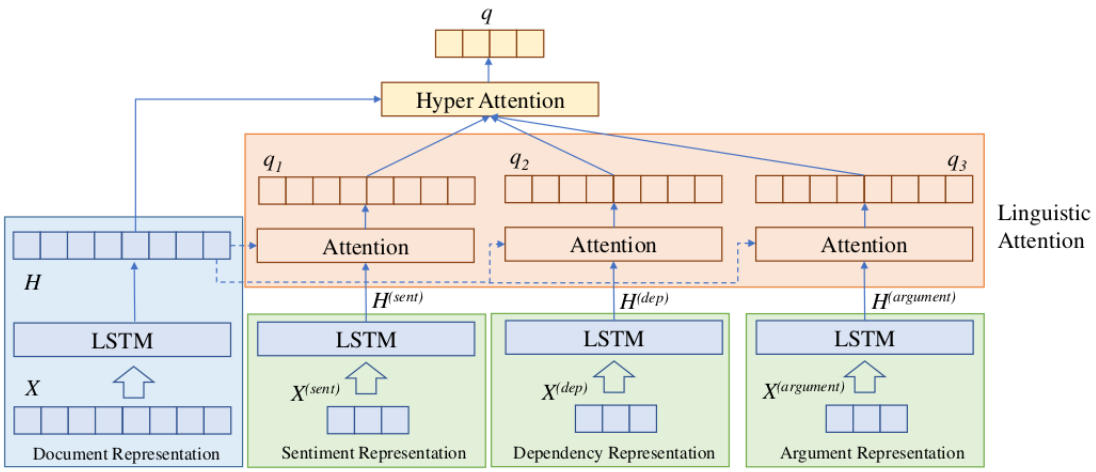


Figure 3.2: Overview of Sun et al. [2018] model.

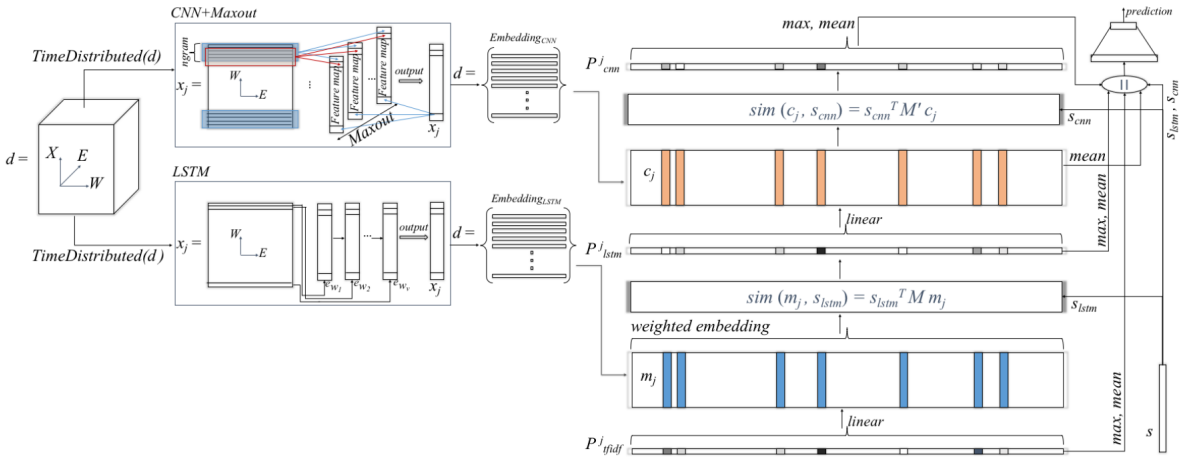


Figure 3.3: Architecture of Mohtarami et al. [2018] for the stance classification task.

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 Multi Dataset Learning (MDL)(multi-dataset learning) has a significant impact on increasing model robustness.

Chapter 4

Methodology

We have used different ideas and methods in this project. In this chapter methods are explained in following *Preprocessing*, *Word Representation*, *Features*, *Machine Learning*, *Balancing*, *Deep Learning*, *Article to Claim*, and *Fake News Detection* sections.

4.1 Preprocessing

The first and mandatory preprocessing step is to tokenize words in the corpus in order to remove and detect special words. After tokenization, a list of punctuation and Persian stop-words are considered as *denied* words and are removed from the corpus in this step. We chose stop-words carefully in a way not to lose refuting or supporting expressions in news context. Verbs carry valuable information to detect stance of an article. Nonverbal stop-word class is a better choice to remove low-value words in this task. Also English number characters will remove from the corpus before tokenizing.

4.2 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.

4.2.1 Similarity

The similarity score is offered by Zarharan et al. [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.

- Ratio: Similarity score in float range 0,1. This parameters calculates from equation 4.1

$$\text{ratio} = \frac{2.0 * M}{T} \tag{4.1}$$

  where:

  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.

4.2.2 Root Distance

This feature is suggested by Zarharan et al. [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.

4.2.3 ImportantWords

List of a controversial and challenging words in news gathered by Zarharan et al. [2019] and considered as *important-words* in this project. 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.

4.2.4 Is Question

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

4.2.5 Has Two Parts

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

4.2.6 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 4.2 average through all similar words polarity scores, to estimate the desired word polarity score.

$$\text{polarity\_score}\,(w) = \frac{\Sigma_{w^{`} \in W} \text{Similatiry}\left(w^{`}, w\right).\text{Polarity}\left(w^{`}\right)}{\Sigma_{w^{`} \in W} \text{Similatiry}\,(w^{`}, w)} \tag{4.2}$$

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. Besides, datasets containing polarity of comments of SnappFood[1] and DigiKala[2] websites are available. 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.

4.3 Balancing

Balancing methods is needed to compensate lack of data in classes except the majority class. Imbalance datasets make models to be biased on the majority class and there may be not enough samples in other classes for a model to learn that. This leads to having high accuracy score (Equation 2.7) while having low f1 score (Equation 2.8). There are several algorithms to deal with imbalanced datasets. It's not practical to rely on only one method and except to perform in the best way. Consequently, three different methods are applied on to the dataset in order to deal with this phenomenon. Methods of balancing a dataset which is used in this project are described in the following sections.

4.3.1 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

[1] snappfood.ir
[2] digikala.ir

from Wikipedia[3] 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 is not enough information in the article about the claim.

According to Figure 5.1 two *Agree* and *Disagree* class in Zarharan et al. [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 Zarharan et al. [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 5.14. Dataset is still imbalanced in one class for both Article-to-Claim (A2C) and Headline-to-Claim (H2C).

4.3.2 Oversampling and Undersampling

Desired oversampling methods to deal with imbalanced dataset are RandomOverSampler, SMOTE, SVMSMOTE, BorderlineSMOTE, and ADASYN which are described in section 2.4.

4.3.3 Tune model 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.

4.4 Machine Learning

Figure 4.1 illustrates a basic schematic of each machine learning models. Input of the model are samples in the dataset. After prepossessing, then predictors extracts from Feature Extractor module. The description

[3] wikipedia.org

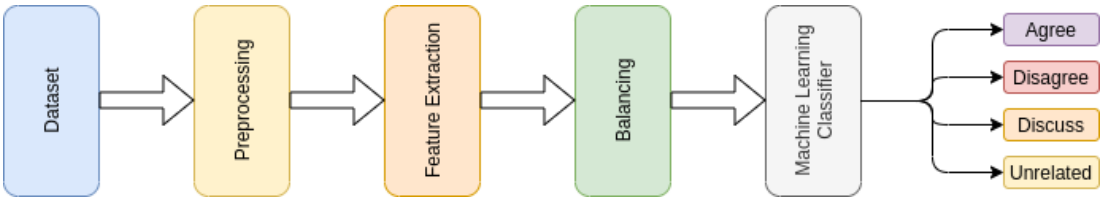


Figure 4.1: Schematic of each machine learning model.

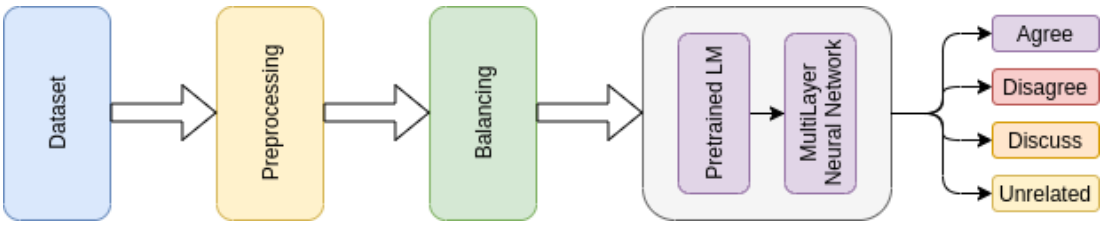


Figure 4.2: Schematic of our deep learning model.

of each predictor described in detail at section 4.2. As the next step, before feeding data into classifier modules, balancing dataset is performed on dataset training set. Finally, samples classify in to one of four desired labels.

4.5 Deep Learning

In the deep learning (DL) approach, a combination of all predictors can be fed into the model and the model on its own will automatically learn which predictor is useful for the task. This property is the biggest advance of DL in comparison to machine learning (P et al. [2021]). On the contrary, in machine learning (ML), it was a critical step to design input predictors that the model can perform the best. And hours of trying different combination of predictors is needed (Giansiracusa [2021]). Schiller et al. [2020] assessed that, In contrast, DL models, ML models that are trained on a single dataset, usually generalize poorly to other domains.

The schematic of the DL model is shown in figure 4.2. H2C and A2C models have the same schematic. Only the input shape of parameters varies in each model. In comparison to Figure 4.1 Feature Extraction is omitted from the ML model. Besides, The deep learning module constructs of pre-trained language model and a few Dense layers at its top. We evaluated BERT, ParsBERT, and ALBERT language models against each other in this project (More description in section 2.5).

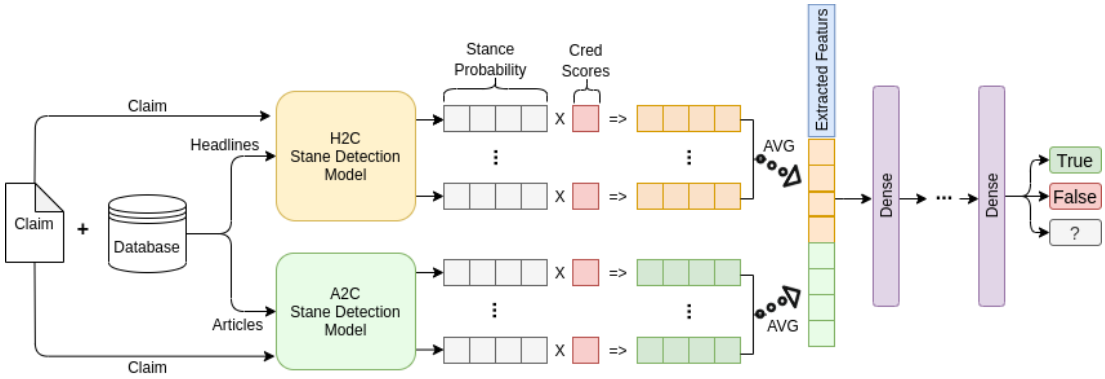


Figure 4.3: Schematic of our fake news detection model.

4.6 Article to Claim

Deep learning models perform far better on language inference tasks so they are better choices for A2C stance classification. To calculate the stance of a claim toward the body of a news article, three different models based on a pre-trained Persian language model based on BERT, ParsBERT, and ALBERT (More description in section 2.5) are evaluated against each other.

4.7 Fake News Detection

Schematic of fake news pipeline is shown in figure 4.3. To detect a news article's veracity, H2C and A2C stance detection models, are considered as a black box. Four news articles are considered to evaluate the veracity of a claim. Firstly, the 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 veracity of the claim.

The credibility of news websites is one of the most important extracted features. The credibility can be calculated through the following steps (The credibility score of H2C and A2C is calculated similarly except using their related stance ground truth):

- **Initialization**: The credibility score of all news websites that are existing in the Zarharan et al. [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, the credibility score is set to 0.1.

- **Quantification:** For each sample in the dataset, the credibility score changes according to Table 4.1. ρ value is calculated from equation 4.3 if it is needed.

Table 4.1: 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

$$\rho = \frac{\mathrm{P}(x, \text{Agree}) - \mathrm{P}(x, \text{Disagree})}{\mathrm{P}(x, \text{Agree}) + \mathrm{P}(x, \text{Disagree})} \tag{4.3}$$

where:

$\mathrm{P}(x, \text{Agree})$ — Probability of Agree

$\mathrm{P}(x, \text{Disagree})$ — Probability of Disagree

- **Score Calculation:** The credibility score for each news website is calculated from equation 4.4.

$$\mathrm{H}(X) = \frac{\sum_{i=1}^{\mathrm{k}_X} \text{credibility of } x_i}{\mathrm{k}_X} \tag{4.4}$$

where:

X — A news website

k_x — The number of news article in the dataset from x

credibility of x_i — Due to table 4.1

Also other 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.

4.8 Conclusion

In this chapter, we described the utilized methodology used in this project. We started with the corpus preprocessing and continued with feature engineering for machine learning models. Then we described our methods to deal with the imbalanced dataset. Then ML, DL, and Fake news detection models are described. In the following chapter (Chapter 5) we will discuss each experiment in detail and make a comparison between them.

Chapter 5

Experiments

In this section, experiment setups and the results of each experiment in each step are discussed. Besides, different ideas and algorithms which are described in section 4 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. Finally, the FakeNews model is evaluated depends on best stance detection models.

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 (A2C). 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 Dataset

The first Persian stance detection dataset (Zarharan et al. [2019]) is used in this project. Zarharan et al. [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 5.1. Claims are retrieved from Shayeaat [1] and Fakenews[2] websites. Each sample contains 3

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

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 is not any information about the claim in the article.

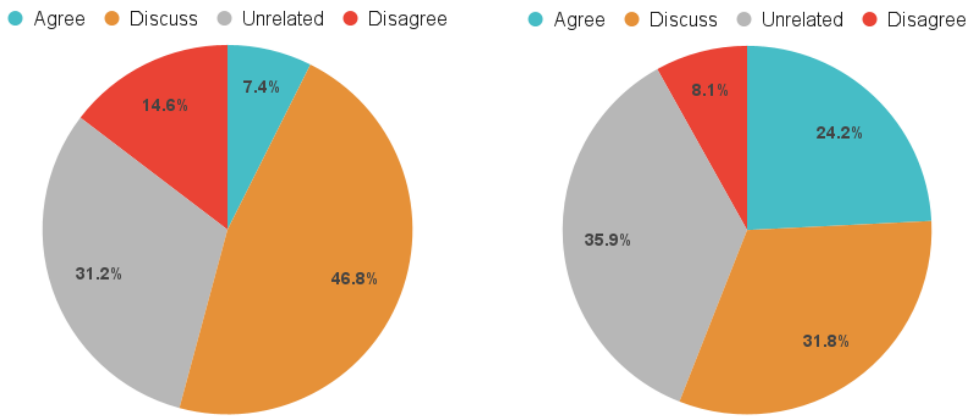
Dataset samples distribution in each class is illustrated in Figure 5.1. According to Figure 5.1, the ratio of *Agree*. Besides, *Disagree* labels are much lower than the others and there is a potential risk for models to be biased on *Unrelated* and *Discuss*. Also, the percentage of *Unrelated* label is higher in H2C than A2C. 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 A2C in comparison to H2C. 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 5.2. In this dataset, the main focus has been on published fake news, this can be inferred from figure 5.2. 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 is not enough integrity between reliable news agencies sources.

Table 5.1: Samples distribution in four classes of Persian stance dataset.

Label	Agree	Disagree	Unrelated	Discuss
H2C	628	210	932	824
A2C	189	374	797	1196



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

Figure 5.1: Comparison between A2C and H2C labels, samples distribution in Zarharan et al. [2019] dataset.

Table 5.2: Fake news data set statistics

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

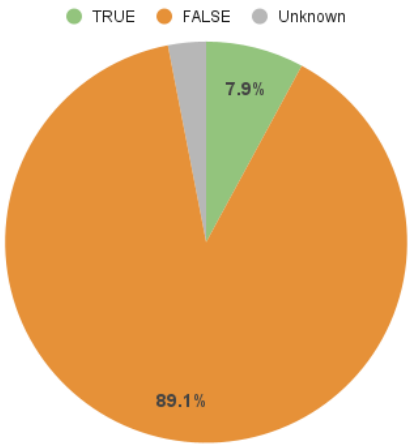


Figure 5.2: Claim veracity label's distribution in Zarharan et al. [2019] dataset.

5.2 Tokenizetion

Tokenization of *Hazm*, *Stanza*, *NLTK*, and *WordPiece* are evaluated against each other manually. Every single difference in each algorithm performance is evaluated[3]. Only three cases are presented as examples to compare each tokenizer's performance in figure 5.3. As we need to remove particular words and patterns from the corpus, 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 5.3, unsuitable tokens are highlighted in blue. There is not any flawless algorithm. WordPiece is subwords based so there are some unknown (Unknown (UNK)) tokens and words are wrongly broken. For example figure 5.3, part (b) blue highlighted word is broken into two pieces wrongly. Considering connected pronouns individually is the strength of *Stanza* tokenizer. But sometimes the Stanza wrongly breaks an original word with a wrong assumption that the desired word has a connected pronoun. Figure 5.3, part (c) is a sample on wrong separating pronoun and at part (b), pronouns are separated correctly. It can be seen in figure 5.3 *Hazm*'s performance is highly similar to *NLTK*. The only difference is that in contrast to *NLTK*, Hazm separates numbers and punctuation in the corpus (Figure 5.3, part (a)).

Besides, the duration of tokenizing for each tokenizer is compared in figure 5.4. While *Hazm* is the fastest words tokenizer among evaluated algorithms, *Stanford* tokenizer lasts significantly longer. According to all pieces of evidence, *Hazm* tokenizer is the best tokenizer for this task.

[3] 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__O##__NG##__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 5.3: Comparison performance of *Hazm*, *Stanza(Stanford)*, *NLTK* and *WordPiece(BERT)* tokenizers.

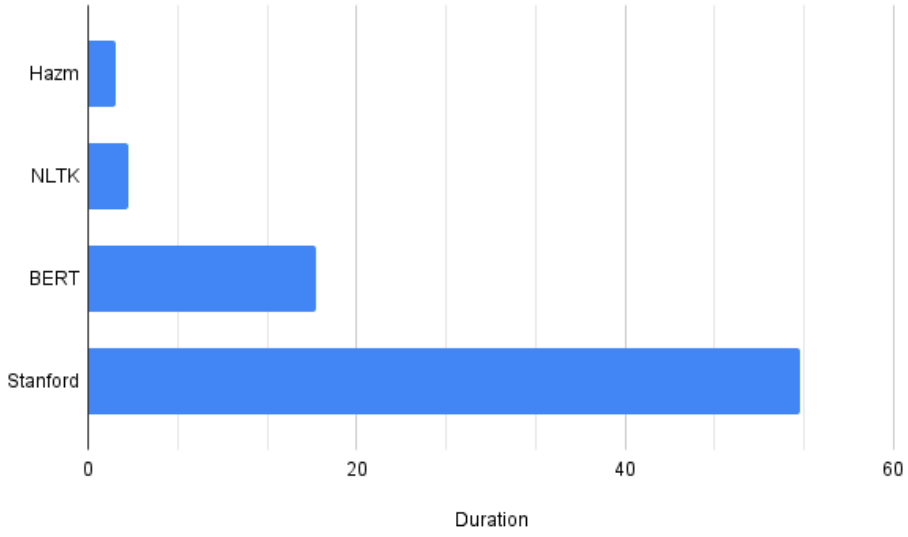


Figure 5.4: Comparison duration of tokenizing algorithm on Zarharan et al. [2019] dataset.

5.3 Stop-Words

After preprocessing tokens, stop-words will be removed from the tokens. Firstly, the same stop-words list which has been used by Zarharan et al. [2019] was used in the project. 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[4] 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, we added and removed some words from Nonverbal list to become suitable for the news context.

The difference between the performance of these four sets of stop-words in addition to skipping removing stop-words (To have a fair comparison) is illustrated in figure 5.5. Zarharan et al. [2019] contains 1255 words which include wide a range of parts of speech. In contrast, NonVerbal[5] stop words set includes 158 words and it doesn't support any verb. In the new version of stop-words, 18 stop words are removed from the Nonverbal 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.

It can be inferred from figure 5.5 that the list of stop-words which is used by Zarharan et al. [2019] is ignoring valuable data and it is even better not to remove stop-words from the corpus. 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. Then all remaining tokens will be concatenated with a space character and considered as prepossessed and clean corpus.

5.4 Word Representation

As a baseline, three different Bag-of-word (Harris [1954]), TF-IDF (Sammut et al. [2010]), and Word-to-Vector (Mikolov et al. [2013]) algorithms are evaluated against each other. More details are explained in

[4] github.com/kharazi/persian-stopwords
[5] Gathered by Kharazi

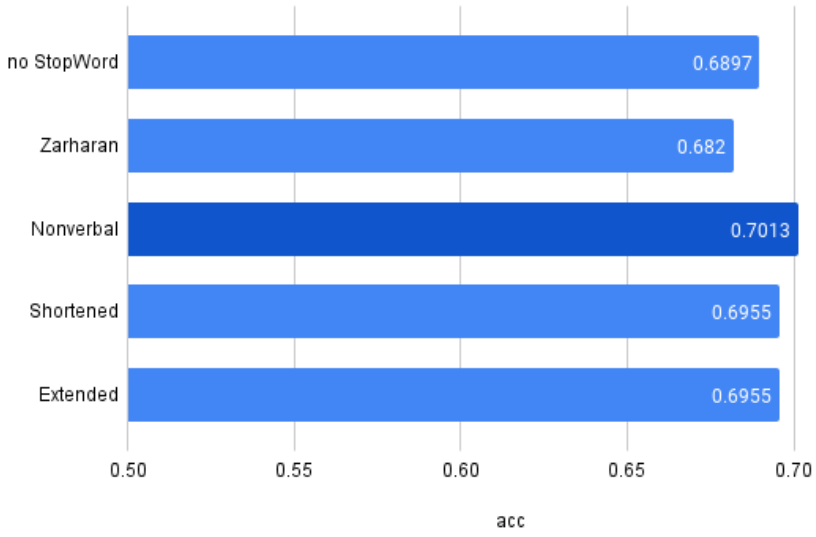


Figure 5.5: Comparison accuracy of SVM model with different configuration of stop words.

section 2.2.

In this project, FastText[6] W2V model is used with vector lengths equal to 300 which is trained on the Persian Wikipedia website. For both TF-IDF and BoW n-gram range is set from 1 to 2.

BoW (Harris [1954]), TF-IDF, and W2V (Mikolov et al. [2013]) performances are compared by the SVM ML algorithm in the stance detection task. According to figure 5.6 TF-IDF performs the best in comparison to BoW and W2V in order to represent words.

5.5 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 (Ho [1995]) results are considered in evaluation to have a more accurate analysis. The similarity score is calculated by utilizing *difflib*[7] python library.

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,

[6] fasttext.cc
[7] docs.python.org/3/library/difflib.html

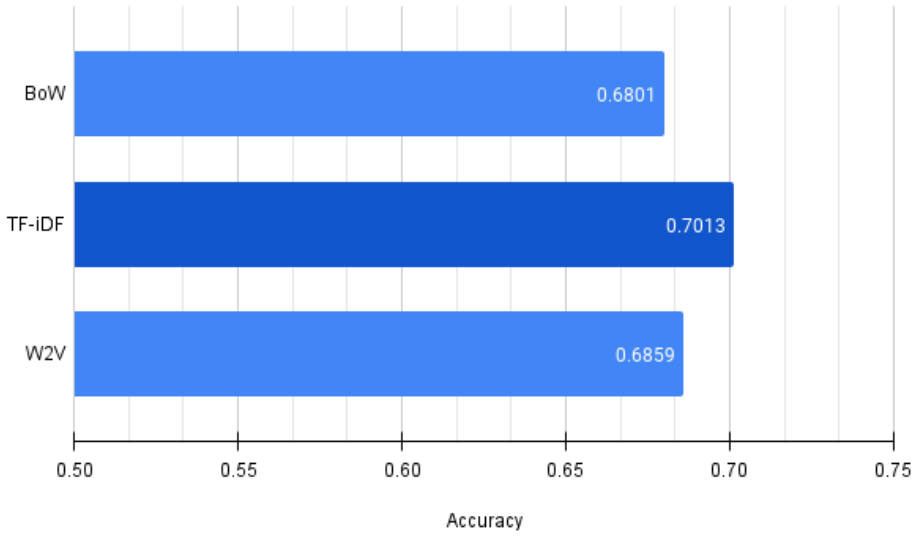


Figure 5.6: Comparison SVM model with BoW, TF-IDF and W2V word representaion algorithms.

all features together are fed to models. According to table 5.3, Similarity and ImportantWords have the highest positive effect respectively, on both accuracy (2.7) and f1-score (2.8). Similarity score has improved accuracy 15 to 17 percent and ImportantWords has improved SVM accuracy by almost 4 percent. Though, 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 5.3, 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 has a negligible effect or improves accuracy.

According to previous comparisons and evaluation inferred from table 5.3, the best predictors to use for stance detection task is the combination of Similarity, ImportantWords, IsQuestion, HasTwoParts, and Polarity predictors in addition to TF-IDF as the corpus representer, altogether.

Table 5.3: Comparison of Acc. score 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

5.6 Machine Learning

In this section, each Gaussian Naive Bayes, SVM, Linear Support Vector Classification (SVC), Random Forest, and Logistic Regression parameters are tuned on stance detection task with respect to chosen predictors in the previous section (5.5). In the next step, the performance of models is compared to each other.

5.6.1 SVM

In this project, We adopted SVC model implementation from *scikit-learn*[8] python library is used in this project. *class_weight* parameter set to *balanced* to compensate imbalanced data. Three SVM classifier tuned parameters are Kernel, Regularization parameter (C) and degree of polynomial kernel. Evaluated kernels are Radial Basis Function (Radial Basis Function (RBF)), Polynomial and Sigmoid. According to figure 5.7, Sigmoid works weak for this task. Each polynomial kernel behaves differently due value of the regularization parameter. Polynomial with degrees 2 and 3 perform better than 1 and 4. Linear polynomial may be so simple and SVM is not good enough to classify stance with a polynomial with degree 4. RBF, Polynomial degree 3 behave similarly due to the regularization parameter changes.

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

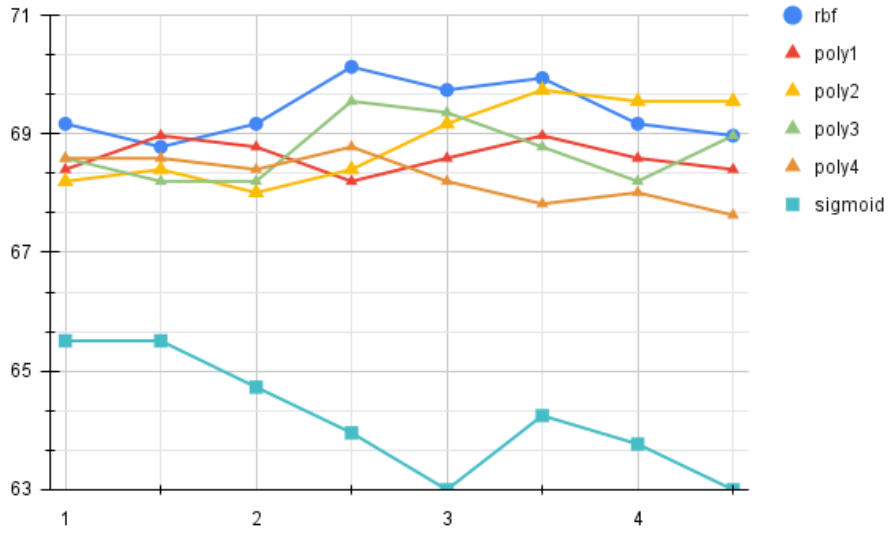


Figure 5.7: Tuning SVM model parameters with TF-IDF representation algorithms

The best configuration for SVM classifier is using RBF as the kernel with regularization parameter equal to 2.5 confirming figure 5.7. Learning procedure and details of each model training exist in this project GitHub repository[9].

5.6.2 Linear SVC

Linear Support Vector Machine Classifier algorithm, loss function and Regularization parameters are tuned with penalty equals to *l2*. Figure 5.8 illustrates comparison between LinearSVC models with loss functions 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 5.8 Best configuration is for Hinge loss and Regularization parameter equals to 1.0.

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

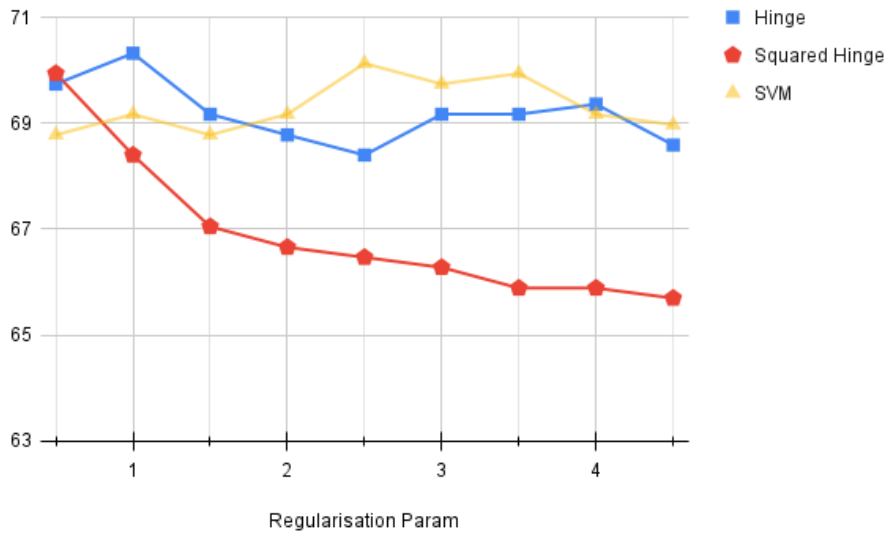


Figure 5.8: Tuning LinearSVC model parameters with TF-IDF representation algorithms.

5.6.3 Random Forest

In this project, implemented Random Forest algorithm from *scikit-learn*[10] 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 the quality of splits in nodes) are tuned for the desired task.

Figure 5.9 illustrates the effect of the number of trees in the forest on accuracy among different configuration. The average accuracy score of models increases by adding more trees into the forest. Besides, *gini* algorithm performs better than *entropy* to measure the quality of splits. Also, three different upper bounds are considered for number of features when looking for a split. No boundary, *sqrt* of total features and *log2* of total feature. According to figure 5.9 don't apply any boundary leads to better accuracy on average. Furthermore, as the boundary gets tighten average performance decreases.

The best configuration for Random Forest ML model in this task is using *gini* algorithm to evaluate splitting quality, not applying any boundary on the number of features and having 125 decision trees in the forest.

[10] scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

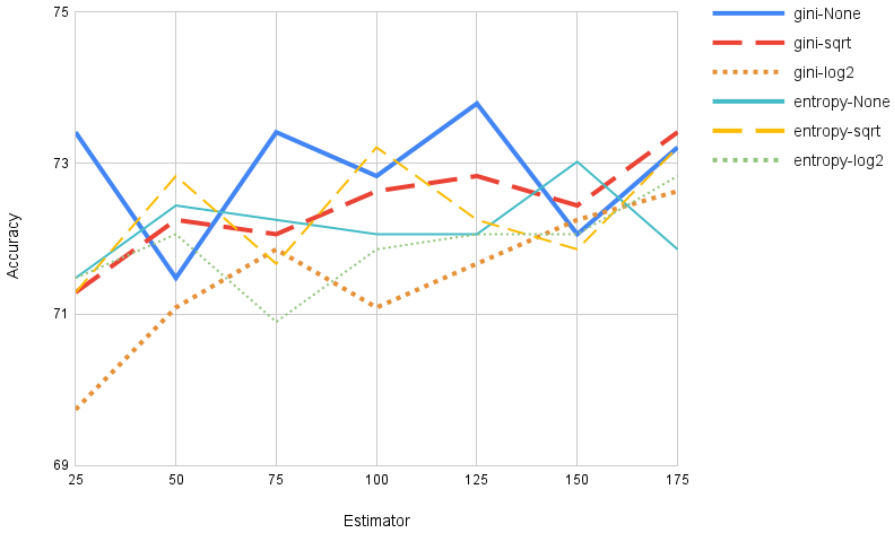


Figure 5.9: Random Forest ML model different configuration on stance detection task. Type of line presents type of the boundary applied on each model. Solid line, dash line and doted line stands for no boundary, sqrt of total feature and log2 of total feature respectively.

5.6.4 Logistic Regression

We also adopted the Logistic Regression algorithm from the *sckiti-learn*[11] python library. Many experiments are designed to evaluate behavior of Logistic Regression models. *Elasticnet* (Equation 2.6) 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 the portion of using $l1$ to $l2$ penalty in *SAGA* solver. Figure 5.10 illustrate the effect of ρ values from 0 to 0.9 on the accuracy of stance detection. Besides, models with a regression parameter from 0.5 to 4.5 are evaluated from the determined ρ range. It can be inferred from figure 5.10 that ρ parameter doesn't have a significant effect on the 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 5.11 which illustrates the effect of the regression parameter on stance classification accuracy. Models with different values of ρ behave similarly and best performances happen when regression parameter is between 1 and 2.5.

Another variant of Logistic Regression setup is to use $l2$ penalty with desired solver algorithms. Figure

[11] scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

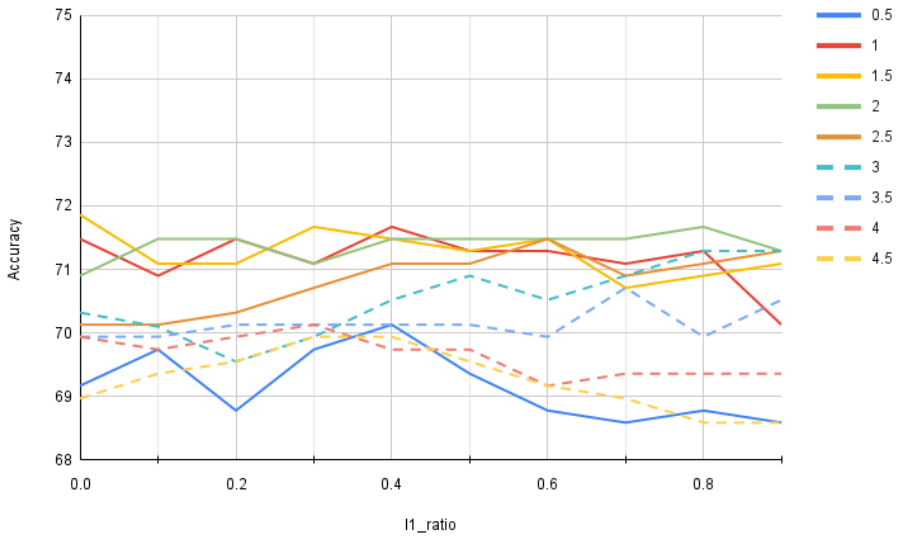


Figure 5.10: Effect of ρ parameter of *elasticnet* penalty on stance detection task.

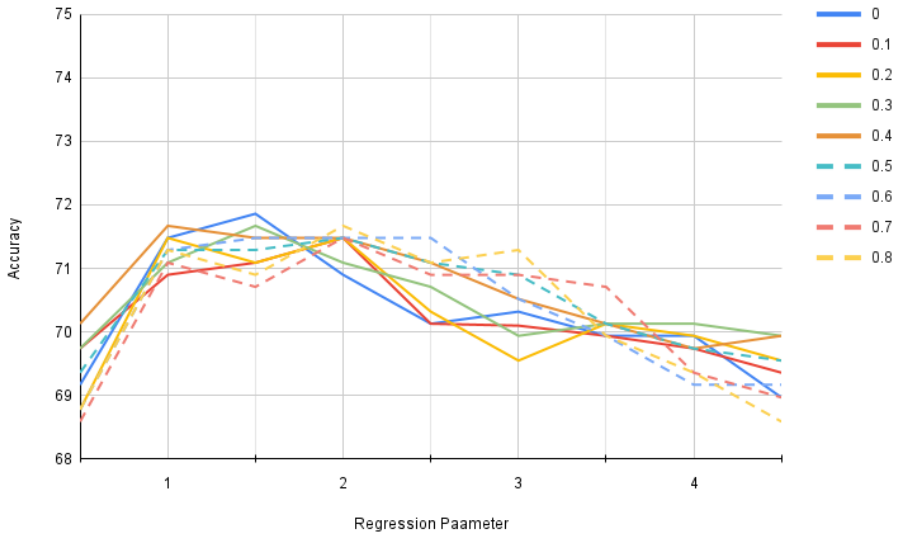


Figure 5.11: Effect of regression parameter of *elasticnet* penalty on stance detection task.

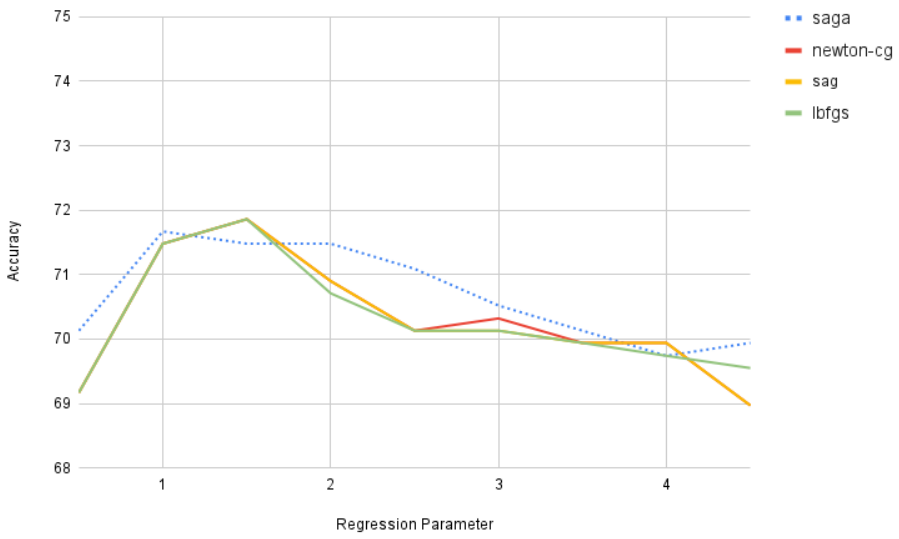


Figure 5.12: Comprasion of Logistic Regression ML models.

5.12 compares best *SAGA* solver with *SAG*, *lbfgs*, and *newton-cg*. The Logistic Regression with *lbfgs* solver and regression parameter equals 1.5 has recorded the highest accuracy.

5.6.5 Comparison

In previous sections parameters of each SVM, LinearSVC, Random Forest, and Logistic Regression models are tuned. In this section models with desired parameters have run 5 times each to have more reliable results and comparison. Average accuracy and highest accuracy recorded in tuning phase, compared in figure 5.13. The highest achievable accuracy with ML models to classify the stance of a claim towards the headline of a news article is 74.01%.

5.7 Dataset Balancing

As mentioned in section Dataset, Figure 5.1, the number of samples in dataset classes was imbalanced. So oversampling should be performed in classes except for the majority class. In Zarharan et al. [2019] minority class forms only 7.4% of data (Figure 5.1). All oversampling methods are evaluated against each other in this project and utilized from the oversampling package of *imblearn* [12] python library.

[12] imbalanced-learn.org/stable/references/over_sampling.html

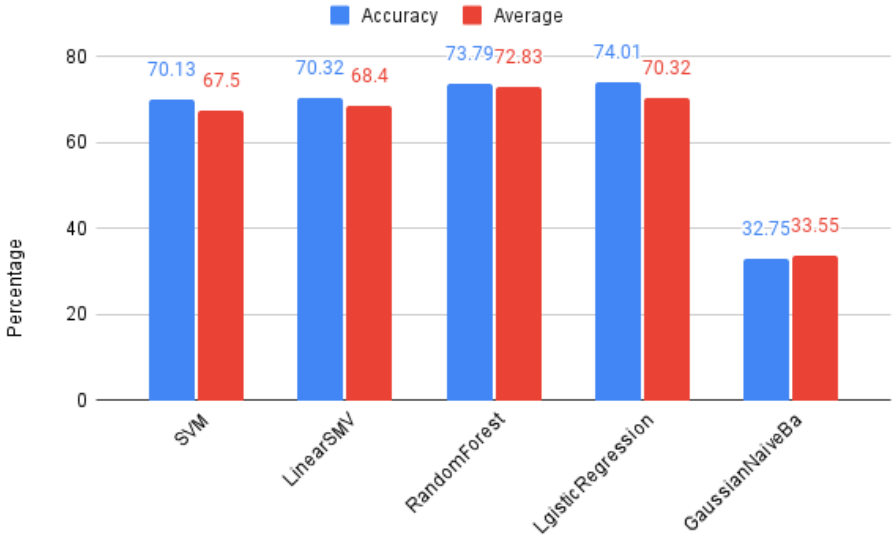
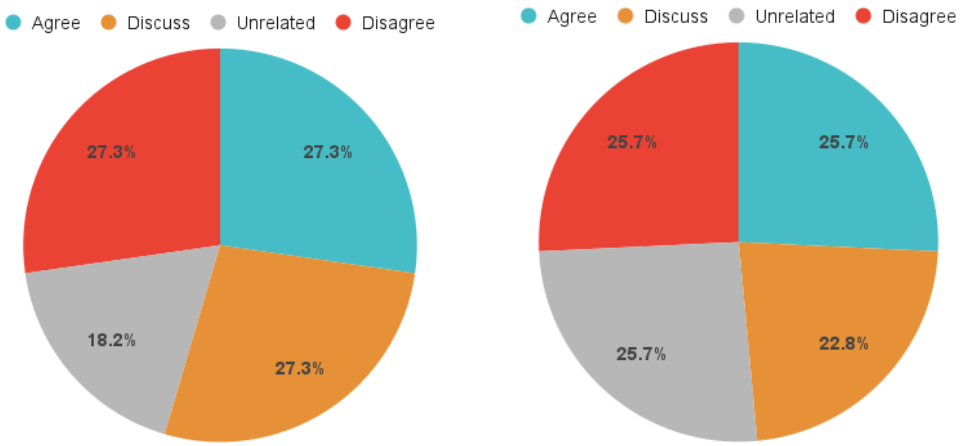


Figure 5.13: Comparison SVM model between BoW, TF-IDF and W2V word representation algorithms.



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

Figure 5.14: Comparison between A2C and H2C labels, samples distribution in Zarharan et al. [2019] dataset, after extending by Zarharan et al. [2021] .

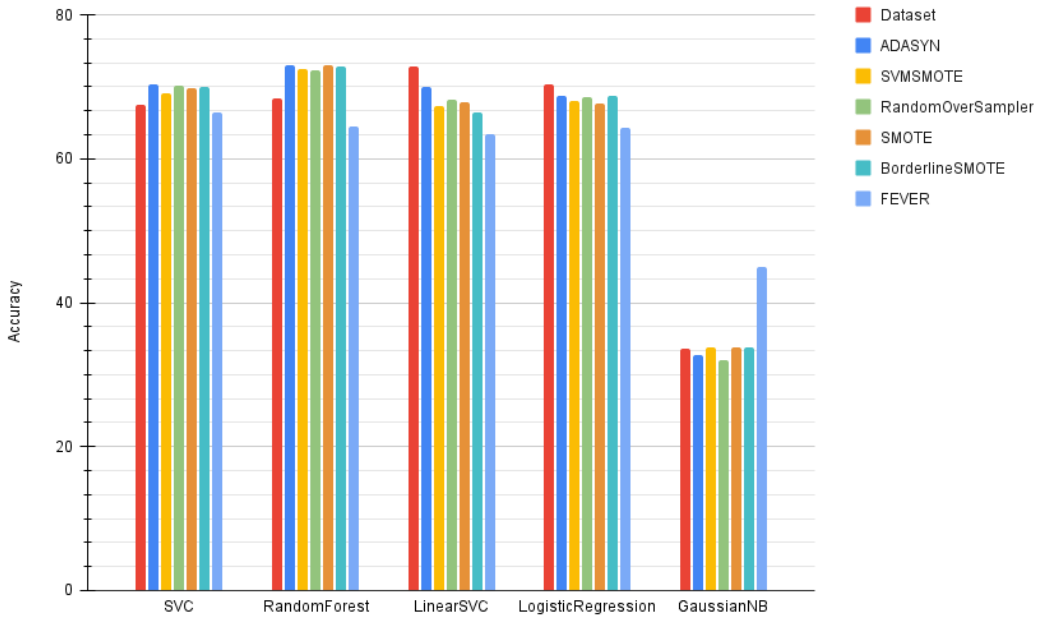


Figure 5.15: Comparison SVM model with BoW, TF-IDF and W2V word representation algorithms.

Firstly, ADASYN, SMOTE, SVMSMOTE, BorderLineSmote, and RandomOverSampler oversampling methods are applied on the Zarharan et al. [2019] dataset. In ADASYN algorithm, the number of nearest neighbors to generate a new sample is set to 9[13]. Each method is evaluated against five desired ML models. Red series in figure 5.15 stands for the accuracy of models, associated with the Zarharan et al. [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.

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

[13] imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.ADASYN.html

Table 5.4: Comparison of H2C stance detection models.

Model	Precision	Recall	F1	Acc.
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**

5.8 Deep Learning

A Pre-trained BERT-based model is used at the top of the model, then two Dense layers and finally a Dense including 4 neurons to classify stance is considered for the end-to-end system. The input of our deep learning (DL) models is *input ids*, *token type ids*, and *attention mask*. BERT (Devlin et al. [2019]), ParsBERT Farahani et al. [2020], and ALBERT (Lan et al. [2020]) models are substitute with Pretrained ML model in figure 4.2. Each epoch lasts about 26 seconds in the training procedure. Figure 5.16 illustrates the model training procedure. In comparison to ML models, the DL model has boosted the accuracy of H2C stance detection by 10 percent.

The BERT-based model has been learned for 20 epochs. Validation loss has been stated to increase since epoch 11 and validation accuracy has not changed considerably then. Best validation accuracy has converged on 80.92% on the H2C dataset.

The ParsBERT-based model has been learned for 20 epochs. Best validation accuracy has converged on 81.11% on the H2C dataset. In comparison to the ML algorithm, DL algorithm has enhanced about 10% accuracy. Besides, the loss score has converged on a lower score than BERT-based model.

Though, the best recorded ALBERT language model on 20 epochs is at most 70.52% on accuracy score. Figure 5.16, parts (e) and (f) is illustrated the training procedure.

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

Figure 5.16: Deep learning procedure on the H2C stance detection task. Left figures illustrate loss score and right figures illustrate accuracy score of train and test data during the 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

Table 5.5: Comparison between A2C ML and DL models.

	H2C		A2C	
Model	F1	Acc.	F1	Acc.
SVM+ADASYN	69.38	70.32	65.18	64.49
ParsBERT	81.96	81.11	78.00	78.32
ParsBERT+ADASYN	85.29	**85.48**	80.29	**80.62**

5.9 Article to Claim

Best stance detection models on both ML and DL models are evaluated on A2C task. The length of 400 characters is considered for the maximum length of article content. Table 5.4 show those models performance on both H2C and A2C task. Using ADASYN oversampler and ParsBERT as the pre-trained language leads to our best results on both H2C and A2C models. We achieved 80.62% accuracy score on the H2C task. Out experiments on A2C achieved a lower score than the H2C task because inferring from longer text is a harder task for the model.

5.10 Fake News

The best BERT-based model for H2C and A2C are considered for this part. These model prediction base on 4 news articles are concatenated with features which are described in section 4.7. Then the overall vector is feed to a three-layer Multilayer-Perceptron model as the classifier. ADASYN oversampling method is also used to deal with imbalanced classes. Figure 5.17 illustrates the 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 5.17, after adding oversampled samples, at first iterations, it is harder for the model to decrease the loss score on the train set and loss score convergent takes a longer time. Though after balancing the dataset (Figure 5.17, part (a)), the value of loss score has converged 0.2 lower than the original dataset (Figure 5.17, part (c)).

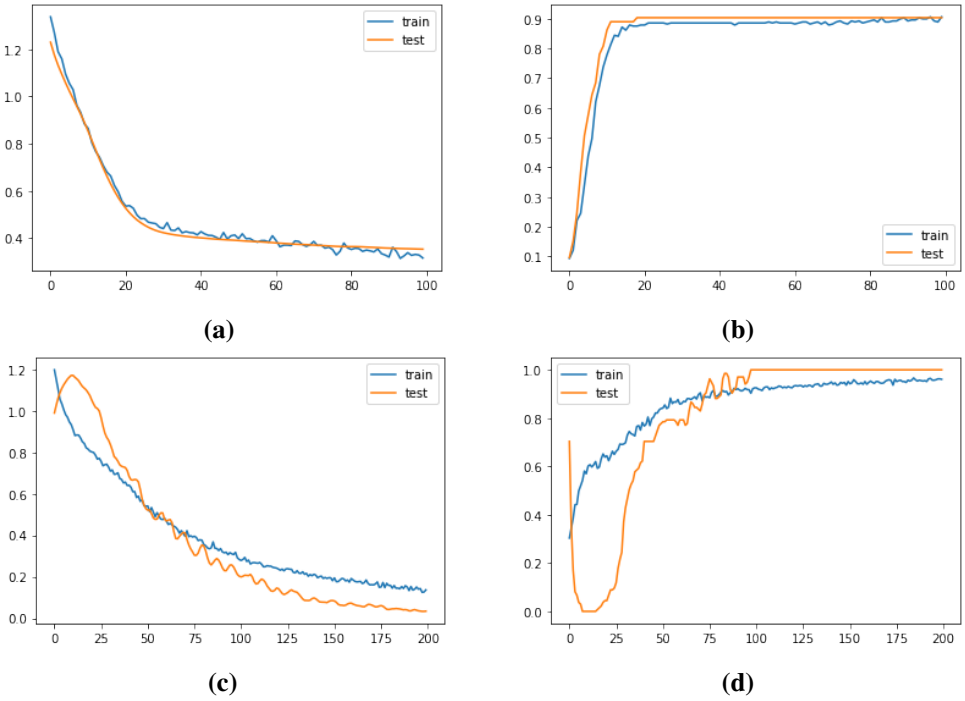


Figure 5.17: Left figures illustrate loss score and right figures illustrate accuracy score of train and test data during training procedure for each iteration. (a, b) Training procedure on fake news detection model trained on the Zarharan et al. [2019] dataset. (c, d) Training procedure on fake news detection model trained on oversampled Zarharan et al. [2019] dataset by ADASYN (He et al. [2008]) algorithm.

Chapter 6

Conclusion

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

In the Persian stance detection task, using deep language models boosts our model performance noticeably. BERT and ParsBERT as its alternative have a high ability to make inferences 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, ParsBERT language models predict stance detection 10% higher than machine learning algorithms 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 A2C task. Despite this issue, machine learning algorithms have achieved an even higher than 70% accuracy score on H2C tasks. 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 H2C and A2C are separately utilized in fake news detection. We have achieved 99% accuracy score on the Zarharan et al. [2019] dataset. Though, the number of samples in the dataset gathered by Zarharan et al. [2019] contains 1624 samples. Increasing number of

samples helps the model to improves its generalization during the training procedure.

It is important to evaluate the fake news detection model performance more precisely. The number of samples in Zarharan et al. [2019] is not enough to achieve a mode with the ability of generalization. Though high accuracy score on the dataset, it is not reliable enough due to the few numbers of samples in the test set.

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

In our fake news detection model, we require A2C and H2C pre-trained models. Even it is not possible for these two models to get fine-tune in fake news detection task. It is worth to taking a look at model architecture and design an end-to-end model for fake news detection. As an alternative, it would be helpful if it is possible for A2C and H2C models to be fine-tuned on the current task at least.

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. volume 2.

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research(AIR), year=2002, volume = 16, pages= 321-357.

Kia Dashtipour, Amir Hussain, Qiang Zhou, Alexander Gelbukh, Ahmad Hawalah, and Erik Cambria. 2016. PerSent: A Freely Available Persian Sentiment Lexicon. In *Advances in Brain Inspired Cognitive Systems*. Springer.

Rahim Dehkharghani. 2019. SentiFars: A Persian Polarity Lexicon for Sentiment Analysis. volume 19, page 12. Association for Computing Machinery.

Liliya Demidova and Irina Klyueva. 2017. SVM classification: Optimization with the SMOTE algorithm for the class imbalance problem. In *2017 6th Mediterranean Conference on Embedded Computing (MECO)*, pages 1–4.

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. arXiv preprint arXiv:1911.11951.

Mehrdad Farahani, Mohammad Gharachorloo, Marzieh Farahani, and M. Manthouri. 2020. ParsBERT: Transformer-based Model for Persian Language Understanding. arXiv:2005.12515.

Noah Giansiracusa. 2021. How Algorithms Create and Prevent Fake News. Springer.

Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. 2005. Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. volume 3644. Springer, Berlin, Heidelberg.

Zellig S. Harris. 1954. Distributional Structure. volume 10, pages 146–162. Routledge.

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. IEEE.

Tin Kam Ho. 1995. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE.

George H. John and Pat Langley. 1995. Estimating Continuous Distributions in Bayesian Classifiers. In *Proceedings of the 11'th 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. International Conference on Learning Representations (ICLR).

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 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.

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 Confer-*

ence 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. arXiv.

Deepak P, Tanmoy Chakraborty, Cheng Long, and Santhosh Kumar G. 2021. Data Science for Fake News: Surveys and Perspectives. volume 42. Springer.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Andreas Müller, Joel Nothman, Gilles Louppe, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*.

Benjamin Riedel, Isabelle Augenstein, Georgios P. Spithourakis, and Sebastian Riedel. 2017. A simple but tough-to-beat baseline for the Fake News Challenge stance detection task. *arXiv:1707.03264*.

Claude Sammut, Webb, and Geoffrey I. 2010. TF–IDF. In *Encyclopedia of Machine Learning*, pages 986–987. Springer.

Benjamin Schiller, Johannes Daxenberger, and Iryna Gurevych. 2020. Stance Detection Benchmark: How Robust Is Your Stance Detection? Springer.

Kai Shu, Suhang Wang, Dongwon Lee, and Huan Liu. 2020. Disinformation, Misinformation, and Fake News in Social Media. Springer.

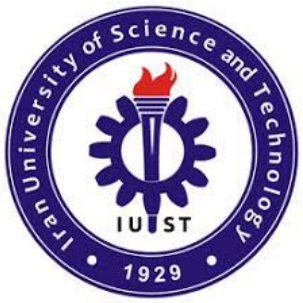
Shivangi Singhal, Rajiv Ratn Shah, Tanmoy Chakraborty, Ponnurangam Kumaraguru, and Shin'ichi Satoh. 2019. SpotFake: A Multi-modal Framework for Fake News Detection. In *2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM)*, pages 39–47.

Qingying Sun, Zhongqing Wang, Qiaoming Zhu, and Guodong Zhou. 2018. Stance Detection with Hierarchical Attention Network. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2399–2409. Association for Computational Linguistics.

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.

Majid Zarharan, Samane Ahangar, Fatemeh Sadat Rezvaninejad, Mahdi Lotfi Bidhendi, Shaghayegh Sadat Jalali, Sauleh Eetemadi, Mohammad Taher Pilehvar, and Behrouz Minaei-Bidgoli. 2019. Persian Stance Classification Dataset. Conference for Truth and Trust Online.

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

School of Computer Engineering