Arabic News Detection

Aya Tamer Ginidy , Shahenda Adel Abdelrahmen ,

Abdulrahman Mohamed Eltahan

I Abstract

Problem Statement:

With the rapid increase in digital news, it's important to classify articles accurately to help users find information easily. This is especially challenging for Arabic news due to the language's complexity. The aim is to develop a system that can accurately classify Arabic news articles into specific categories.

Solution Outline:

This project will use CNN, LSTM, BERT, and AraBERT models to build a high-performing Arabic news classification system. We will use the Akhbarona-News-Classification repository as a starting point and the SANAD dataset for training and evaluation. The project includes several key tasks to ensure thorough development and optimization.

II Introduction

The rapid proliferation of online news content has created a pressing need for effective and efficient news detection systems, particularly in languages with less representation in natural language processing (NLP) research, such as Arabic. With over 400 million native speakers, Arabic is one of the most widely spoken languages in the world. However, the complexity of the language, characterized by its rich morphology, diverse dialects, and script variations, poses significant challenges for NLP applications.

News detection, the process of automatically identifying and classifying news articles, is critical for various applications, including news aggregation, trend analysis, and misinformation detection. In the context of Arabic, the task becomes more challenging due to the scarcity of annotated datasets, the presence of multiple dialects, and the intricate syntactic and semantic structures of the language.

In this paper, we address the problem of Arabic news detection by leveraging recent advancements in NLP and machine learning. We propose a novel approach that combines traditional text processing techniques with modern transformer-based models to accurately classify Arabic news articles. Our contributions include the creation of a comprehensive dataset of Arabic news articles, the application of various machine learning models, and an extensive evaluation of their performance.

III Baseline

III. I Dataset

SANAD Dataset is a large collection of Arabic news articles that can be used in different Arabic NLP tasks such as Text Classification and Word Embedding. The articles were collected using Python scripts written specifically for three popular news websites: AlKhaleej,AlArabiya and Akhbarona.

III. II Preprocessing

This preprocessing pipeline transforms raw text into a cleaner format suitable for analysis or machine learning. It involves standardizing case, removing unwanted characters and spaces, and optionally applying techniques like tokenization, stop words removal, stemming, and lemmatization. Each step helps in reducing noise and improving the quality of the text data. Note that some steps are commented out and might be used based on specific needs or preferences

-Convert text to lowercase.

-Remove punctuations from text:
Example:"مرحبا، بالعالم!" becomes "مرحبا بالعالم"

-Remove references and hashtags from text:

Example:"مرحبا @احمد ، كيف حالك؟" becomes "مرحبا ، كيف حالك؟"

--Tokenization and Removal of Stop Words

• Tokenization: Break the text into individual words or tokens using word_tokenize.

• Stop Words Removal: Filter out common words (stop words) that may not add significant meaning to the text (e.g., "and" "the", "is"). stp.stopwords_list() provides a list of stop words to be removed.

• Joining: Combine the remaining words back into a single string separated by spaces.

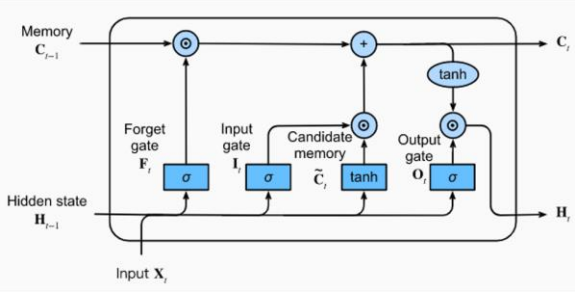
III. III Naive Bayes
Model Selection
With a dataset in place, cleaned and ready for processing,
The next step was transformed text data into a numerical format using the CountVectorizer, which converts the text into a matrix of token counts.
First, we used Naive Bayes

The Naive Bayes classifier provided a solid baseline for the Arabic news detection task. Its performance can be attributed to the probabilistic nature of the model, which effectively handles the high-dimensional feature space typical in text classification.

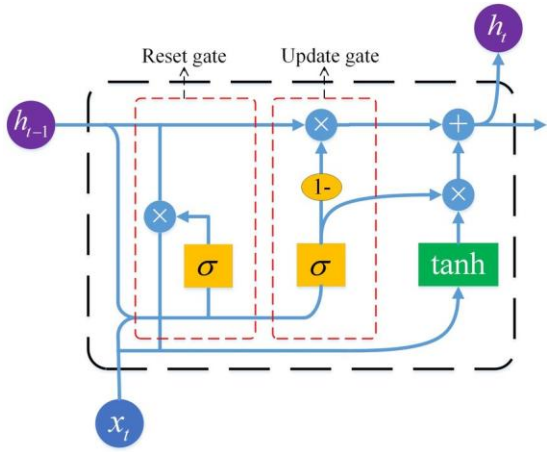
III. IV RNN
RNNs are powerful neural network architectures designed to handle sequential data by maintaining hidden states that capture contextual information.RNN has hidden state mechanism(The hidden state is updated at each time step based on the input and the previous hidden state) allows to retain and utilize contextual information from previous time steps, which is essential for understanding dependencies in sequential data.RNNs can be applied to various types of sequence data, including text, audio, and time series data, making them versatile for many applications. Challenge and Limitations of RNN is suffering from vanishing or exploding gradients, which can hinder learning, especially for long sequences. This problem occurs when gradients become too small or too large, causing instability in the learning process.RNNs may struggle to learn long-term dependencies in sequences due to the vanishing gradient problem. This makes it difficult for the network to retain information over many time steps.Training RNNs can be computationally expensive and time-consuming, particularly for large datasets and long sequences.

III.V LSTM and GRU
1-Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Network (RNN) designed to capture long-range dependencies and patterns in sequential data, addressing the limitations of traditional RNNs.LSTMs introduce a more sophisticated structure called a memory cell, which is equipped with three gating mechanisms: the input gate, forget gate, and output gate. These gates control the flow of information into, within, and out of the memory cell, enabling LSTMs to maintain and update information over long periods.LSTMs can remember information for long periods, making them effective for tasks like language modeling, speech recognition, and time series prediction.Challenge and Limitations of LSTM are computationally intensive and require significant resources to train, especially for large datasets and deep architectures. This can lead to long training times.Due to their complexity, LSTMs can be slower during inference compared to simpler models.LSTMs require

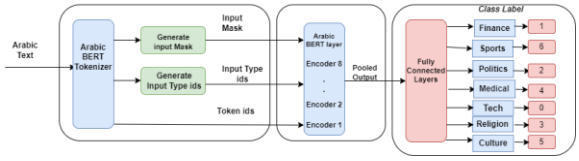
more memory for storing the network parameters and intermediate states compared to simpler models. While LSTMs can handle longer sequences than traditional RNNs, their performance can still degrade with very long sequences. Transformers, on the other hand, can process longer sequences more effectively due to their self-attention mechanism



2-GRUs simplify the LSTM architecture by combining the forget and input gates into a single update gate, reducing computational complexity while maintaining performance.GRUs have fewer gates (two compared to three in LSTMs), making them simpler and faster to compute.Fewer parameters mean faster training and inference times compared to LSTMs.In many tasks, GRUs perform comparably to LSTMs while being more efficient.Challenge and Limitations of GRU is less empirical evidence and fewer studies specifically focusing on GRUs compared to LSTMs, making it harder to understand their performance across all possible scenarios.GRUs do not have a separate cell state (memory) to store long-term information, which might be less effective in some tasks requiring long-term memory



III.VI Transformer-Based Models
Transformer-based models have revolutionized the field of natural language processing (NLP) by providing state-of-the-art performance on various language understanding tasks. Unlike traditional recurrent models, transformers rely on self-attention mechanisms to process input sequences in parallel, capturing long-range dependencies more effectively.



1-BERT (Bidirectional Encoder Representations from Transformers) is a transformer-based model that marked a significant advancement in natural language processing (NLP) by achieving state-of-the-art results in various tasks. Unlike previous models, BERT is deeply bidirectional, meaning it considers the context from both the left and right sides of a word when making predictions, allowing it to understand nuanced word meanings in context. The core architecture behind BERT is the Transformers, which rely on self-attention mechanisms to process text sequences, capturing long-range dependencies more effectively than recurrent neural networks (RNNs).

Traditional models process text unidirectionally (left-to-right or right-to-left), but BERT reads

the entire sequence of words at once, providing a full context for each word. This bidirectional approach enhances contextual understanding.

BERT uses token embeddings, segment embeddings, and positional embeddings to represent each token in the input sequence. The self-attention mechanism allows BERT to weigh the importance of different words in context, capturing dependencies between words regardless of their distance in the sequence. BERT consists of multiple layers of transformers, with each layer applying self-attention followed by a feed-forward neural network.

Challenges and limitations of BERT include:

- Significant computational resources are required for both pre-training and fine-tuning, often necessitating specialized hardware like GPUs or TPUs.
- BERT can be slower in inference compared to simpler models, making it less suitable for real-time applications where speed is crucial.
- BERT's performance is sensitive to fine-tuning settings; small changes in hyperparameters, training duration, or dataset characteristics can lead to significant differences in performance.

2-RoBERTa (Robustly Optimized BERT Pre-training Approach) is a variant of the BERT model developed by Facebook AI to enhance performance and address certain limitations of the original BERT model. Key improvements in RoBERTa include:

- Removal of Next Sentence Prediction (NSP): Unlike BERT, RoBERTa eliminates the NSP task, as it was found not to contribute significantly to performance.
- Longer Training Sequences: RoBERTa uses sequences up to 512 tokens during training, capturing more context within a single sequence for better performance on downstream tasks.
- Larger Training Corpus: RoBERTa is pre-trained on a significantly larger corpus, including datasets like Common Crawl News, Web Text, Wikipedia, and Books Corpus, totaling over 160GB of uncompressed text data.
- Improved Training Techniques: RoBERTa uses larger mini-batches and a dynamic learning rate schedule with warm-up periods, stabilizing the training process and improving model convergence.

RoBERTa works similarly to BERT but with different hyperparameters. Challenges and limitations include:

- Inference Speed: Despite better performance, RoBERTa's larger model size and increased complexity can result in slower inference speeds, which can be a limitation for real-time applications.
- Computational Resources: The improvements come at the cost of increased computational resources, requiring more GPU/TPU resources and longer training times compared to BERT, making it less accessible for those with limited resources.

3-DistilBERT is a smaller, faster, and cheaper version of BERT, introduced by Hugging Face through "knowledge distillation." It aims to retain most of BERT's performance while being more efficient. Key aspects of DistilBERT include:

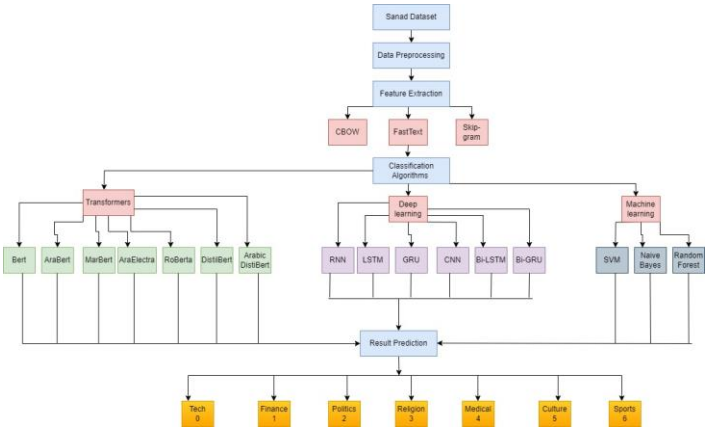
- Reduced Parameters: DistilBERT reduces the number of parameters by approximately 40% and speeds up inference time by about 60%.

- Simplified Architecture: It achieves this by reducing the number of layers from 12 (in BERT-base) to 6, while maintaining the same hidden size, vocabulary size, and sequence length as BERT-base.
- Triple Loss Function: DistilBERT uses a combination of standard cross-entropy loss, distillation loss, and cosine embedding loss.

Challenges and limitations of DistilBERT include:

- Specialized Tasks: It may not perform as well as BERT when fine-tuned for highly specialized tasks.
- Generalization: In some cases, DistilBERT might not generalize as well as BERT, especially on tasks requiring deep contextual understanding.
- Nuanced Information: Due to its reduced size and layers, DistilBERT may not capture as much nuanced information as BERT, leading to slightly lower accuracy on some tasks.

IV Extensions



In our preprocessing, we removed the step to convert text to lowercase. especially since we are working with an Arabic dataset.

-Normalize Arabic text:
Original: "إأآا باىىء, مؤمن, شئ, لغة, كتاب"
After normalization: "اااا باىيء, ءمن, شء, لغه, كتاب"
-Remove diacritics
Original: "اللُّغَة العَرَبِيَّة تُكْتَب بِحُرُوف مِعْجَمِيَّة"
After removing diacritics: " اللغه العربيه تكتب بحروف معجميه"

IV. I Word Embedding
I. Skip-gram Overview:
Skip-gram is a training algorithm used by Word2Vec to predict context words surrounding a given target word. Here's a brief overview:

- Objective: The model aims to predict the context words that appear within a specified window around a target word.
- Training: It updates word vectors so that the target word's vector becomes closer to the vectors of its context words, maximizing the probability of context words given the target word.

Pros of Skip-gram

- Effective for Rare Words: Performs well with rare words and learns useful representations for less frequent words.
- Contextual Information: Captures detailed context, aiding in understanding word meanings and relationships.

- High-Quality Embeddings: Produces embeddings that capture semantic and syntactic similarities between words.

Cons of Skip-gram

- Training Time and Resources: Computationally expensive and time-consuming, especially with large vocabularies and datasets.
- Increased Memory Usage: Requires significant memory to store and update numerous word vectors.
- Difficulty with Large Datasets: May struggle with efficiency and scalability for extremely large datasets.
- Hyperparameter Sensitivity: Performance is sensitive to hyperparameters like window size, vector size, and training epochs, requiring careful tuning.

Overall, Skip-gram provides high-quality word embeddings and captures detailed context, but it can be resource-intensive and sensitive to hyperparameter settings.

II. CBOW

CBOW (Continuous Bag of Words) is another training algorithm used by Word2Vec, which focuses on predicting a target word based on its surrounding context words. Here's a brief overview:

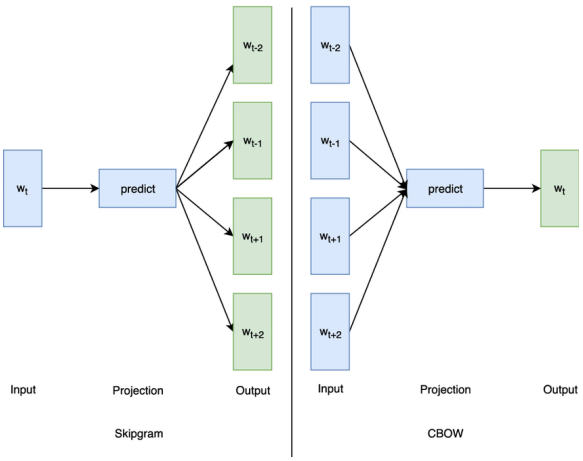
- **Objective**: The model aims to predict a target word given the context words (the words surrounding it) within a specified window size.
- **Training**: CBOW learns word representations by adjusting the word vectors so that context words' vectors come together to predict the target word, optimizing the probability of the target word given the context words.

Pros of CBOW:

- **Faster Training**: Compared to the Skip-gram model, CBOW is computationally more efficient because it averages context words to predict the target word, leading to faster training times.
- **Better for Frequent Words**: CBOW tends to perform well with frequent words, as it averages the context, which can smooth out noise in larger datasets.
- **Reduced Memory Usage**: Since it processes multiple words at once (by averaging), CBOW generally requires less memory compared to Skip-gram.

Cons of CBOW:

- **Less Effective for Rare Words**: CBOW tends to underperform on less frequent words, as it focuses on predicting a word from multiple surrounding words and might not learn detailed representations for rare words.
- **Contextual Loss**: By averaging the context words, CBOW might lose some important information about the order and relationships between words in the context.
- **Quality of Embeddings**: While faster, CBOW often produces slightly lower-quality embeddings compared to Skip-gram, especially when dealing with infrequent words.



skip gram & cbow

III.FastText

FastText Process Overview:

- **Objective**: FastText aims to learn word embeddings by considering both the entire word and subword (character-level) information. This allows it to capture better semantic representations, especially for morphologically rich languages like Arabic.
- **Training**: FastText works similarly to the Skip-gram model but uses character-level n-grams in addition to whole words. This helps the model generate embeddings for rare words and those that may not appear in the training corpus by building them from their subword units.

Pros of FastText:

- **Effective for Rare and Out-of-Vocabulary Words**: By using character-level n-grams, FastText can generate embeddings for words not present in the training data, which is particularly useful for rare words or words with slight spelling variations.
- **Morphologically Rich Languages**: FastText performs well with languages that have complex word forms or inflections (e.g., Arabic), since it learns from subword units and captures the structure of words.
- **Faster Training**: FastText generally trains faster than models like GloVe or traditional Word2Vec because it processes subword n-grams, which allows the model to learn more efficiently.

Cons of FastText:

- **More Memory Usage**: Since FastText deals with character n-grams in addition to word-level embeddings, it requires more memory to store and manage the embeddings.
- **Training Time**: While FastText can produce high-quality embeddings, it can still be computationally expensive due to the additional subword processing, especially when applied to large datasets with long sequences.

IV. II Data Augmentation

1. **Sampling for Deep Models**

- **Purpose**: This strategy takes a **fixed number** of samples (6000) per label, selecting the first available rows for each label. It is useful when you want to **balance** the dataset by ensuring each class has the same number of samples, but without any randomness.
- **Use Case**: This is suitable for **deep models**, like CNNs or RNNs (e.g., LSTM), where the dataset size is large, and balanced sampling is critical for model performance.
- **Strengths**:
- Simple and fast.

- Ensures each label has the same number of samples, which helps avoid class imbalance issues.
- **Limitations**:
- If the dataset is ordered (i.e., earlier samples might differ in characteristics from later ones), this might not represent the entire data distribution well.

2. **Sampling for Transformers**

- **Purpose**: This strategy takes a **random sample** of rows (1000 per label), allowing for a more representative and diverse selection from the dataset. This is critical for models that are sensitive to the diversity of the data, like transformers.
- **Use Case**: Transformers (e.g., BERT, DistilBERT) often require more careful sampling to ensure the model learns from various contexts and avoids overfitting to patterns that may exist if data is ordered. Random sampling provides diversity.
- **Strengths**:
- Introduces **randomness**, ensuring that the model sees diverse examples from the dataset.
- Especially useful for transformer-based models, which can benefit from exposure to a wider range of examples.
- **Limitations**:
- If the random sampling is not reproducible (i.e., random_state is not set), you might get different samples each time, leading to slightly different training results.

IV. III CNN

Convolutional Neural Networks (CNNs) are a powerful tool in deep learning, particularly for image-related tasks, due to their ability to automatically and efficiently extract features from raw data.

Architecture of CNNs:

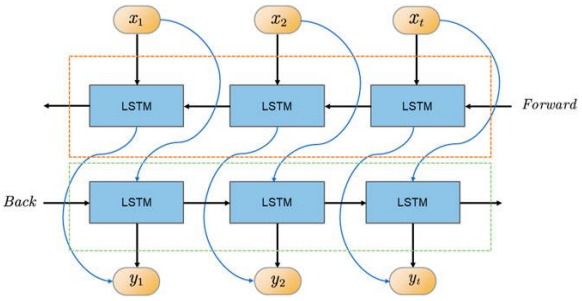
1. Convolutional Layer: The core building block, consisting of learnable filters (kernels) that slide over the input data. Each filter performs a convolution operation, producing a feature map that highlights specific patterns in the data.
2. Activation Function: Typically ReLU, applied after convolution to introduce non-linearity, enabling the network to learn more complex patterns.
3. Pooling Layer: Used to downsample feature maps, reducing their spatial dimensions while retaining important information. This helps reduce computational complexity and prevent overfitting.
4. Fully Connected Layers: After several convolutional and pooling layers, high-level reasoning is performed by fully connected layers. These layers flatten the feature maps into a single vector and pass it through one or more dense layers for classification or regression tasks.

Challenges and Limitations of CNNs:

- Large Amount of Labeled Data: CNNs require a large amount of labeled data for training to achieve high performance, which can be challenging to obtain in some domains.
- Computational Intensity: Training CNNs, especially deep ones, is computationally intensive and requires powerful hardware like GPUs.
- Interpretability: The complex nature of CNNs makes it difficult to interpret how they make decisions, which can be a concern in applications requiring transparency.
- Overfitting: If not properly regularized, CNNs can overfit to the training data,

leading to poor generalization on unseen data.

IV. IV BI-LSTM - BI-GRU
1-Bi-directional LSTM (Bi-LSTM) A Bi-directional LSTM (Bi-LSTM) processes input sequences in both forward and backward directions, capturing context from past and future states. This approach is useful for various sequence-based tasks like text classification, named entity recognition, and time-series forecasting.

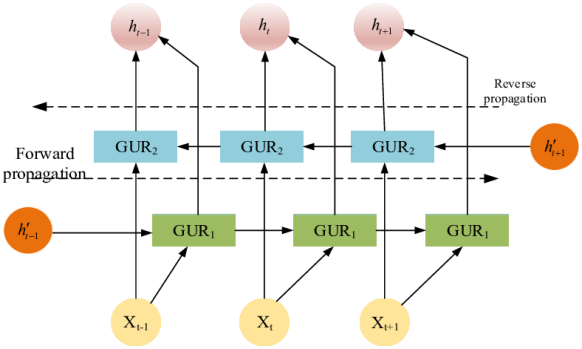


- Architecture:
    - Forward LSTM: Processes the sequence from beginning to end.
    - Backward LSTM: Processes the sequence from end to beginning.
    - Combining Outputs: Concatenates outputs from both directions at each time step for a richer sequence context.

Challenges and Limitations:

- Increased Parameters and Computation: Processing in both directions doubles the number of parameters and computations, leading to longer training times and higher resource consumption.
- Memory Usage: Storing both forward and backward hidden states increases memory usage, which can be a bottleneck for long sequences or large datasets.

2-Bidirectional GRU (Bi-GRU) A Bidirectional GRU (Bi-GRU) extends the basic GRU by processing the input sequence in both forward and backward directions, giving the model access to both past and future context. This is useful for tasks similar to those addressed by Bi-LSTM.



- Architecture:
    - Forward GRU: Processes the sequence from beginning to end.
    - Backward GRU: Processes the sequence from end to beginning.
    - Combining Outputs: Concatenates outputs from both directions at each time step for a richer sequence context.

Challenges and Limitations:

- Increased Parameters and Computation: Like Bi-LSTM, Bi-GRU doubles the computational workload compared to a unidirectional GRU, leading to higher computational costs and longer training times.
- Memory Usage: The need to store forward and backward hidden states increases memory usage.

Common Factors:

- Both Bi-LSTM and Bi-GRU process sequences in both directions, capturing richer context.

- Both face challenges with increased computational costs, training times, and memory usage due to their bidirectional nature.

Conclusion: Bi-LSTM and Bi-GRU provide richer sequence context by processing input in both directions, beneficial for tasks requiring comprehensive sequence understanding. However, they come with trade-offs in terms of increased computational workload and memory usage. Careful consideration is needed based on application requirements and available resources.

IV.V SVM
Support Vector Machine (SVM)

After establishing a baseline with Naive Bayes, we moved on to using Support Vector Machines (SVM), which is a powerful discriminative classifier. SVM works by finding the hyperplane that best separates the different classes in a high-dimensional feature space, which is particularly well-suited for text classification problems with sparse data.

For this task, we utilized the transformed text data (using CountVectorizer) to train the SVM model. The model's ability to maximize the margin between classes helps it generalize well to unseen data, making it a strong candidate for classification tasks where feature spaces are large, like in text analysis.

Additionally, by experimenting with different kernels (linear, RBF), we could adjust the SVM to better fit the structure of the data, which led to improved performance over the Naive Bayes baseline in detecting Arabic news categories.

IV.VI Random Forest

Following our experimentation with Naive Bayes and Support Vector Machine (SVM), we employed Random Forest, an ensemble learning method that builds multiple decision trees during training and merges their results to improve accuracy and control overfitting.

Random Forest works by randomly selecting features and training multiple decision trees, each of which produces a class prediction. The final prediction is made based on the majority vote across all the trees, which adds robustness to the model. This approach is particularly useful in text classification tasks like Arabic news detection, where the dataset may contain noisy or irrelevant features.

The ensemble nature of Random Forest helped mitigate the risk of overfitting and allowed the model to handle the high-dimensional feature space generated by CountVectorizer. It provided strong performance by capturing complex patterns in the data while maintaining interpretability, making it a solid choice for improving classification accuracy over the baseline models.

IV. VII Arabic language models

MARBERT (Arabic BERT)

- Description: MARBERT is a variant of the BERT model fine-tuned for the Arabic language. It leverages large Arabic text corpora to pre-train a model that understands the nuances and complexities of Arabic, including its complex morphology, diverse dialects, and rich linguistic structures.
- Architecture: Uses the same architecture as BERT but is trained with Arabic text.
- Advantages: Captures context from both directions in the text, leading to a deeper understanding of word meanings and relationships.

AraBERT

- Description: AraBERT is a transformer-based language model designed to handle Arabic text. It is based on Google's BERT architecture, adapted for the specific needs of the Arabic language.
- Disadvantages:
    - Computational Resources: Requires significant computational resources and a large amount of text data.
    - Complexity: Transformer models are often complex and hard to interpret, making it challenging to understand why the model makes certain predictions.

AraELECTRA

- Description: AraELECTRA adapts the ELECTRA model specifically for the Arabic language.
- Advantages: Utilizes a different pre-training approach compared to BERT, focusing on distinguishing real input tokens from corrupted ones, which can lead to more efficient training.

Arabic DistilBert

Arabic DistilBERT is a distilled version of BERT (Bidirectional Encoder Representations from Transformers), specifically fine-tuned for the Arabic language. DistilBERT was developed by Hugging Face as a smaller, faster, and more efficient version of BERT while maintaining around 97% of its performance

Disadvantages:

While Arabic DistilBERT is a powerful and efficient model, it has some disadvantages:

•Performance Trade-off: Although it retains 97% of BERT's performance, there is still a slight drop in accuracy, especially in more complex or nuanced language tasks.

•Reduced Contextual Understanding: Due to the reduced number of layers, the model may not capture as deep contextual relationships as the full BERT model, which could affect its performance on tasks requiring intricate language understanding.

Common Challenges and Considerations

- Computational Resources: Training large transformer models like MARBERT, AraBERT, AraELECTRA, and Arabic DistilBert
- requires significant computational power.
- Interpretability: These models are often complex and hard to interpret, which can be a drawback in applications requiring transparency.
- Bias in Training Data: The performance of these models is highly dependent on the quality and diversity of the training data. Biases in the data can lead to biased outputs.

Summary: Arabic language models like MARBERT, AraBERT, AraELECTRA, and Arabic DistilBert
leverage advanced transformer architectures to better handle the unique aspects of the Arabic language. While they offer improved performance in understanding and generating Arabic text, they also share common challenges such as high computational resource requirements, complexity, and potential biases in training data.

V Results

V. I Baseline Results

Model	Accuracy
RNN	0.578
LSTM	0.938
GRU	0.937
Naïve Bayes	0.83
BERT	0.90
DistilBERT	0.89
RoBERTa	0.78

V.II Word Embedding Tries on LSTM

Types	Accuracy
Skip_gram	0.8799422979354858
CBow	0.9116666913032532
FastText	0.9284523725509644

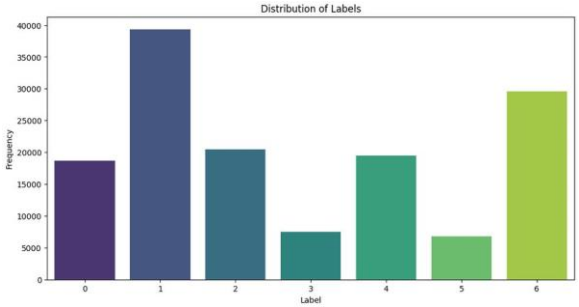
V. III Our Extension: with the modified preprocessing function and FastText

Model	Test Loss	Accuracy
RNN	1.0792728662490845	0.6253571510314941
CNN	0.2160913646221161	0.932023823261261
LSTM	0.23052658140659332	0.9284523725509644
GRU	0.19021539390087128	0.9438095092773438
BiLSTM	0.234378680586814 88	0.9258333444595337
BiGRU	0.19409875571727753	0.9358333349227905
Naïve bayes	----------	0.8551190476190477
SVM	----------	0.934047619047619
Random Forest	----------	0.9195238095238095

V. IV transformers

Model	Training Loss	Validation Loss	Accuracy	F1	Precision	Recall
BERT	0.435000	0.574921	0.815714	0.814938	0.814551	0.815714
DistilBERT	0.554700	0.779104	0.730000	0.729026	0.729483	0.730000
AraElectra	0.142900	0.163163	0.944761	0.944977	0.954711	0.955820
RoBERTa	0.848800	1.003361	0.641429	0.632415	0.639512	0.641429
AraBERT	0.245900	0.237062	0.945714	0.945724	0.946714	0.945714
MaraBERT	0.110300	0.289157	0.941429	0.941540	0.942433	0.941429
Arabic DistilBERT	0.125700	0.260642	0.927143	0.926858	0.929532	0.927142

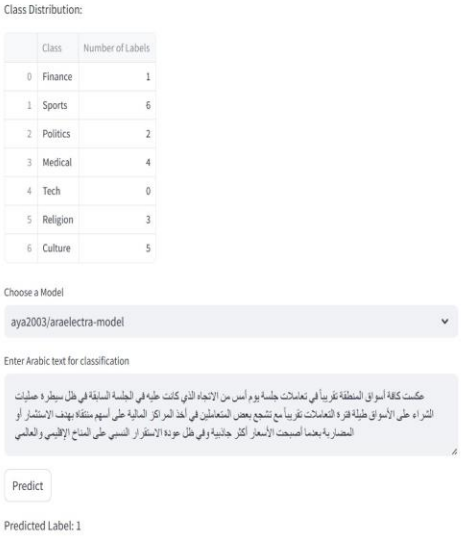
V.V Analyzing the data



After analyzing the data, we found that it is unbalanced. The solution is to perform data augmentation.

VI Application

In our application, we utilize four specific Transformer models—AraBERT, AraELECTRA, MARBERT, and Arabic DistilBERT—due to their superior accuracy in comparison to other transformers. We have saved and loaded each of these models using Hugging Face, ensuring seamless integration and prediction capabilities within the application,



We made our application using Streamlit Streamlit application for Arabic news detection is an exciting project that combines natural language processing (NLP) with an interactive web interface to provide real-time insights into Arabic news content. The goal of the Streamlit application is to detect and classify Arabic news articles based on their content. This could involve tasks like categorizing news into different topics (e.g., Finance, medical, Religion, Culture,politics, sports, technology). It includes functions to load models and predict text labels, a user interface for selecting models, and displaying class distribution. Users input Arabic text and receive classification results based on their model choice.

VII Future work

- **Expanding Dataset Coverage**: Increasing the variety of news sources and incorporating more diverse datasets to capture a broader range of dialects and topics.
- **Real-time Application**: Implementing techniques to speed up inference time to

make the system usable for real-time news classification, such as further optimization of models or the use of more efficient transformer models like DistilBERT.

- **Multilingual Integration**: Integrating the system with other languages commonly spoken in the region to create a multilingual news classification tool.
- **Misleading Information Detection**: Extending the system to detect misinformation and fake news by incorporating more complex models like GPT for content generation and comparison.
- **Hybrid Models**: Experimenting with hybrid models that combine CNNs or RNNs with transformer-based approaches to optimize performance while reducing computational costs.

VIII Conclusion

This project successfully developed an Arabic news classification system, leveraging various machine learning and deep learning models. By integrating CNN, LSTM, BERT, and AraBERT models, the system was able to address the unique challenges posed by the Arabic language, such as its rich morphology and diverse dialects. The use of the SANAD dataset and advanced preprocessing techniques like normalization and tokenization helped improve the quality of the input for the models, enhancing overall performance.

The best results were obtained using transformer-based models, particularly AraELECTRA and AraBERT, which excelled in capturing the contextual meanings of words in Arabic. These models outperformed traditional approaches due to their ability to process long-range dependencies and complex linguistic features of the language. However, the computational cost associated with these models and the need for fine-tuning were notable challenges.

In addition to building the models, a user-friendly application was developed to make the news classification process more accessible. This application allows users to input Arabic news articles and obtain immediate classification results, demonstrating the practicality and real-world usability of the system. The app's interface makes it easy for non-experts to interact with the underlying complex machine learning models, turning a technical solution into an intuitive tool for end-users.

VIII References

1.SANAD Dataset

2.akhbarona dataset

3. Arabic Fake News Detection