NLP

SENTIMENT ANALYSIS OF ARABIC TEXT

USING TRADITIONAL, NEURAL, AND TRANSFORMER-BASED NLP MODELS

22110377
NLP
YARA AL-HARAHSHEH

PREPARED BY MERA ALSALHI

Arabic Sentiment Analysis

Project Idea & Real-World Motivation

Project Overview

The goal is to build a **sentiment analysis system** specifically tailored for Arabic text, classifying emotions into Positive, Neutral, and Negative categories to enhance understanding.

Real-World Application

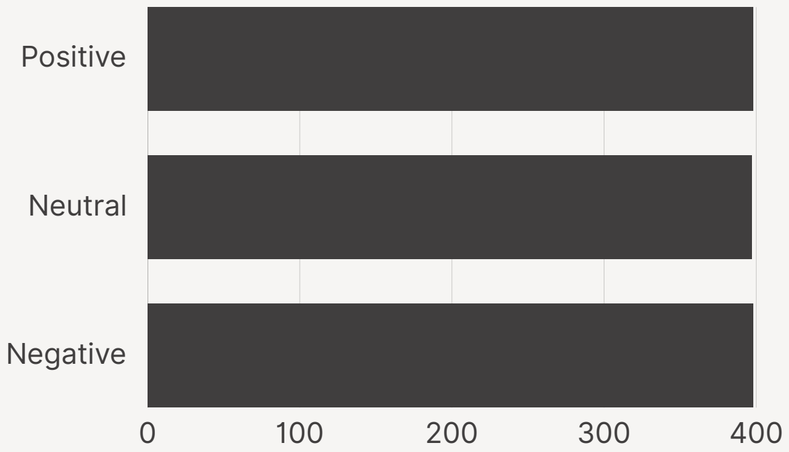
This system can be utilized for **social media opinion analysis**, allowing businesses to gauge public sentiment, analyze customer feedback, and monitor market trends effectively and efficiently.

Why Arabic NLP?

Arabic language presents unique challenges due to its **morphological richness** and high dialectal variation, necessitating the implementation of advanced NLP techniques to achieve accurate results.

Dataset Description

DATASET OVERVIEW



OBSERVATIONS

- ARABIC DATASET FOR SENTIMENT CLASSIFICATION
- TOTAL SAMPLES 1193
- DATASET IS BALANCED
-

Preprocessing Techniques

Strategies for Text Preparation

P0 – Raw Text

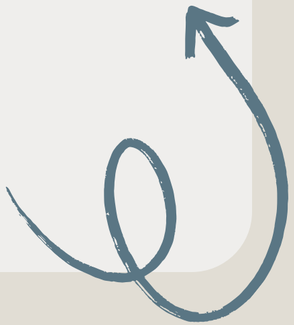
The original text was utilized without any cleaning, ensuring that all initial data was intact for analysis and review, providing a baseline for comparison against processed versions.

P1 – Regex Cleaning

This method involved the removal of URLs, mentions, emojis, and symbols from the text, streamlining the content to focus solely on relevant textual information for better clarity.

P2 – Text Normalization

Techniques such as Arabic letter normalization, lowercasing, and noise reduction were applied to improve text consistency, enhancing the quality of data before any further analysis or processing.



Evaluation of Preprocessing Methods

RAW TEXT PERFORMANCE

- Using raw text resulted in **lower performance** due to noise, including URLs and mentions, which impacted consistency in Arabic forms.

REGEX-BASED CLEANING

Regex-based cleaning significantly enhanced results by removing **irrelevant patterns**, thus reducing noise and improving overall text quality for analysis.

WHY NORMALIZATION WORKED BEST

- it reduced arabic spelling variation for example the different form of Alef
- also it reduced sparsity in TF-IDF features
- it helped all models generalize better

Evaluation of Preprocessing Methods

Task	Library Used	Explanation
Text Cleaning	re (regex)	I used regex to remove noise such as URLs and special characters efficiently.
Tokenization & Stopword Removal	nltk	I used NLTK because it provides Arabic stopwords and reliable tokenization.
Lemmatization	Not applied	I did not apply lemmatization because contextual embeddings in AraBERT already capture word morphology.
Normalization & Diacritics Removal	AraBERT preprocess	I used AraBERT preprocessing to ensure consistency with transformer-based embeddings.

Word representation comparison

Representation	How it Works	Strengths	Limitations	Performance
TF-IDF	Represents text as term-frequency statistics	Simple, fast, interpretable	Cannot capture semantics or word order	Lowest performance
FastText (avg)	Uses pre-trained word vectors averaged per sentence	Captures subword information	Loses context and word order	Moderate performance
AraBERT (CLS)	Contextual embeddings from transformer	Understands context and polysemy	Computationally expensive	Highest performance

numerical comparision table

Embedding	Model	Accuracy	Macro-F1
TF-IDF	SVM	~0.64	~0.64
FastText	GRU / BiLSTM	~0.53	~0.48
AraBERT (fine-tuned)	Transformer	0.69	0.69

Traditional Word Representation: TF-IDF

what i used

- i used TF-IDF to represent text as frequency

Strengths

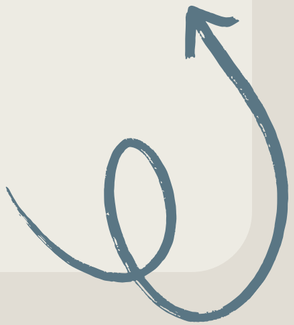
- Simple and computationally efficient
- Performed well with linear models such as SVM
- Served as a strong baseline in my experiments

limitation

- Does not capture semantic meaning
- Ignores word order and context
- Vocabulary-dependent

Observation

- TF-IDF performed best when combined with text normalization and SVM



fastText embeddings

what i used

- i used pre-trained FastText embeddings and averaged word vectors to represent sentences

Strengths

- captures morphological information using subword modeling
- better than TF-IDF for arabic word forms

limitation

- context-independent
- sentence representation loses word order information

Observation

- FastText performed worse than TF-IDf in my experiments when used with SVM

Dependent Pre-trained Embeddings: AraBERT

what i used

- i used AraBERT CLS embeddings to represent sentence level meaning
- later, i fine tune araBERT as a full transformer model

Strengths

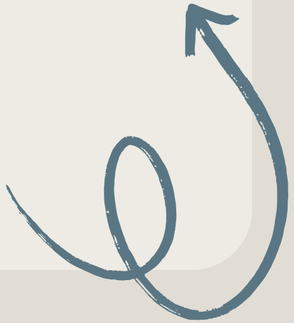
- captures contextual meaning and semantics
- handles ambiguity in Arabic effictively
- Support for Long Sequences:

limitation

- computationally expensive
- requires fine tuning to achieve best performance
- ocabulary Limitations and OOV Tokens

Observation

- Fine-tuned AraBERT achieved the highest overall performance in my project.



Experimental Flow

Experimental Flow Followed in My Project
To follow the instructor's guidelines, I structured my experiments in stages

stage A: Pipeline Comparision

- I fixed one baseline model (SVM).
- I evaluated:

1. (3 preprocessing methods)
2. (3 embedding strategies)

- This resulted in 9 experiments only

stage B: Best pipeline selection

- i selected the best pipeline based on Macro-F1 score

stage C: Model Family Comparision

- I fixed preprocessing and embeddings
- i compared:

1. traditional ML models
2. Neural networks
3. Tranformer-based models

stage A results stage B choosing the best pipeline

Baseline model: SVM

From the 9 experiments, I identified the best-performing pipeline:

- **Preprocessing: Text normalization (P2)**
- **Embedding: TF-IDF**
- **Model: SVM**

reason:

- **this pipeline achieved the highest Macro-F1 score**

Export as file | Refresh data | Report an issue | 9 rows x 5 columns | Go to column | Viewing

	Preprocess	Embedding	BaselineModel	# Accuracy	# MacroF1
	Missing: 0 (0%)	Missing: 0 (0%)	Missing: 0 (0%)	Missing: 0 (0%)	Missing: 0 (0%)
	Distinct: 3 (33%)	Distinct: 3 (33%)	Distinct: 1 (11%)	Distinct: 7 (78%)	Distinct: 8 (89%)
	P0 Raw 33%	TF-IDF 33%	SVM 100%	Min 0.502092050... Max 0.640167364...	Min 0.500617259... Max 0.640270153...
	P1 Regex 33%	FastTextAvg 33%			
	P2 Norm 33%	AraBERT-CLS 33%			
0	P0 Raw	TF-IDF	SVM	0.602510460251046	0.6025320591358326
1	P1 Regex	TF-IDF	SVM	0.602510460251046	0.6025320591358326
2	P2 Norm	TF-IDF	SVM	0.6401673640167364	0.6402701539123883
3	P0 Raw	FastTextAvg	SVM	0.5648535564853556	0.563958662761058
4	P1 Regex	FastTextAvg	SVM	0.5648535564853556	0.5640509725016768
5	P2 Norm	FastTextAvg	SVM	0.5271966527196653	0.527185078557067
6	P0 Raw	AraBERT-CLS	SVM	0.5104602510460251	0.5079463814772377
7	P1 Regex	AraBERT-CLS	SVM	0.502092050209205	0.5006172591311909
8	P2 Norm	AraBERT-CLS	SVM	0.5564853556485355	0.5555456108728531

Stage C: Traditional ML Models

Traditional Machine Learning Models:

Models I compared

- SVM
- Logistic Regression
- Decision Tree

Best traditional model

- SVM
- Achieved the highest Macro-F1 score
- Showed strong generalization

Stage C: Neural Network Models

Neural Network Models

Models I implemented

- GRU
- BiLSTM

Observation

- Neural models capture sequential information
- BiLSTM performed better than GRU
- Performance was lower than traditional ML due to dataset size

Stage C: Tranformer Model

Transformer-Based Model: Fine-Tuned AraBERT

- **what i did**
- I fine-tuned AraBERT end-to-end for sentiment classification.
- I fine-tuned AraBERT end-to-end for sentiment classification.
- **Results**
- Achieved the highest Accuracy and Macro-F1.
- Outperformed all other model families.

Stage C: Tranformer Model

Transformer-Based Model: Fine-Tuned AraBERT

- **what i did**
- I fine-tuned AraBERT end-to-end for sentiment classification.
- I fine-tuned AraBERT end-to-end for sentiment classification.
- **Results**
- Achieved the highest Accuracy and Macro-F1.
- Outperformed all other model families.

Final Model Comparision

Model Family	Best Model	Accuracy	Macro-F1
Transformer	AraBERT Fine-tuned	0.69	0.69
Traditional ML	SVM	0.64	0.64
Neural Network	BiLSTM	0.53	0.48

Key Findings & Takeaways

- **Preprocessing matters for Arabic**

Normalization and diacritics removal improved results across all models.

- **Word representation choice is critical**

TF-IDF ignores meaning, while AraBERT captures context and ambiguity.

- **Model complexity improves performance**

Traditional ML < Neural Networks < Transformers.

- **Fine-tuned AraBERT performed best**

Achieved the highest Macro-F1 on the sentiment classification task.

- **Macro-F1 was used instead of accuracy**

To fairly evaluate performance across all sentiment classes.

THANKYOU