King Saud University
College of Computer and Information Sciences
Information Technology Department

IT469
Human Language Technologies

Topic Detection

Prepared by

Group#: 3	
Leader Email: 437200221@student.ksu.edu.sa	
Amal Mohammed Alotibi	437201331
Haifa Yousef Alkhudair	437200221
Reem Yahya Mohammed	437200008
Waad Khalid Alsaleh	437203086

Supervised by
Dr. Nora Madi

Roles and Responsibilities:

Student name	Role	Reviewed section	Correction made
Amal	Introduction	All Sections	-
Haifa	Data	All Sections	-
Reem	Text Cleaning and preprocessing	All Sections	-
Waad	Feature engineering and modeling	All Sections	-
All	Evaluation	-	-
All	Cover page and figures references	-	-
All	Coding	-	-
All	Deployment		

Table - 1: Roles and Responsibilities

Table of Contents

List of Tables

List of Figures

1. Introduction

The general topic for the project is Topic Detection. Topic detection is a Natural Language Processing (NLP) task used to determine the topic of a given text. It is useful when we have a handful of texts and we want to classify which topic belongs to which text. For example, if we are interested in a certain topic and we want to extract texts from social media websites regrading that topic we use the Topic Detection task [1]. Another application for Topic Detection is in customer service, it helps in the process of determining the type of issue or problem the customer is reporting about [1].

Nowadays data is being generated in large numbers and volumes. And this data can hide a lot valuable information that we can extract knowledge from. And with these large numbers it is hard for humans or almost impossible to scan through them and detect each topic for each document, so, using NLP tasks to do this job is more efficient and time saving. Topic Detection can be used in various fields, such as: business, medical, education, mass communication and news.

This project aims to develop a model that detects the topic of a certain text using machine learning algorithms and NLP techniques and tasks. This project follows the general NLP framework, starting from Data Acquisition.

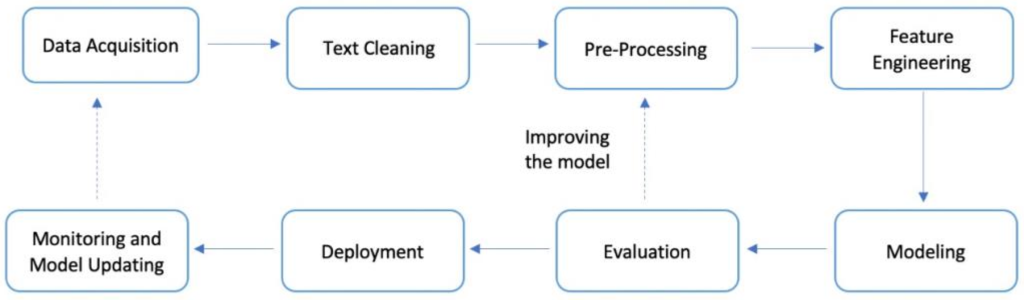


Figure - 1: NLP Framework Illustration

The purpose of this report is to showcase the steps of performing the Topic Detection task using Python programming language and other beneficial libraries and packages. In this report, we are going to provide details about each step in the NLP framework and how we applied it to this specific problem. In the following sections, we are going to describe the data used, how it was prepared to be used in the models, how the models were made as well as evaluating those models.

2. Data

“The dataset has nine categories each of which contains 300 documents in Arabic Language. Each category has its own directory that includes all text files belonging to this particular category.”[1] The categories of the documents are one of the following: Art, Economy, Health, Law, Literature, Politics, Religion, Sport and Technology (9 Classes).

The dataset has five versions:

- **version1:** the original dataset.
- **version2:** the dataset after removing stop words, punctuation and diacritics.
- **version3:** version 2 after applying the light10 stemmer.
- **version4:** version 2 after applying Chen stemmer.
- **version5:** version 2 after applying Khoja algorithm for extracting the roots.

Using this dataset, we are aiming to train models that predict the topic of a given text.

To extract the dataset contents, we defined a function that receives the text files and extract each text then it stores it in a list that will be cleaned and preprocessed in the next stage.

3. Text Cleaning and Preprocessing

After taking a general look on the dataset and comparing the differences between each version, we decided to perform the cleaning and preprocessing tasks on all versions to ensure that they are ready to be trained in the models.

Text Cleaning

For this task, we defined a function that receives the text and cleans the following (if found):

1. URLs and emails.
2. Mentions (incase the text was a tweet).
3. Emoticons and emoticons encodings.
4. Arabic diacritics.
5. English letters (since the text is in Arabic).
6. Phone numbers.
7. Punctuation marks and symbols.

We also unified some Arabic Letters (e.g. إ آ أ were unified to ا), translated English numbers to Arabic numbers, removed newline marks (\n) that resulted from reading the file and Right-to-Left mark Unicode (\u200f).

[1] This information was taken from the dataset description file attached with it (readme.txt).

A model that gets trained on uncleaned data will not provide accurate results and has a poor performance, hence the cleaning step is necessary.

Preprocessing

To preprocess our data, we did the Tokenization task as well as Tokenization with Part-of-Speech task (to compare two different preprocessing approaches) using MADAMIRA. MADAMIRA is "a system for morphological analysis and disambiguation of Arabic" [2].

We defined a function that extracts the cleaned datasets from the lists and store them inside text files so we can pass them as raw inputs to the MADAMIRA system.

After the preprocessing, the output from the MADAMIRA system was as follows:

- For the Tokenization only task, the output was a text file that has all the articles and enclitics (all clitics) tokenized[2].
- And for the Tokenization with Part-of-Speech, the output was a text file that has each word with its morphological feature attached to it.

Morphological features from MADAMIRA do not only include the part-of-speech feature but also other closed-class features such as the gender and case[3].

The last step was reading the files after the preprocessing and we did that using a defined function that extracts and reads a text file and stores it in a list.

4. Feature Engineering

Since our model predicts the topic of a text, the feature we used is the topic and it has 9 possible values that corresponds to the categories from the dataset. Pandas library from python was used to store the text lists into DataFrames (A structure similar to a table) where we have 2 columns. The (text) column which has the text body and the (topic) column which indicates the category. In this step, we obtained 5 DataFrames one for each version of the dataset and these DataFrames will be used to train the model in the next step.

[2] This information was taken from the MADAMIRA v2.0 User Manual
[3] This information was taken from the MADAMIRA v2.0 User Manual

5. Modeling

Tf-idf / AraVec

Tf-idf is a statistical measure used for creating word embeddings. It was applied to all the dataset members before splitting the dataset for training and testing.

AraVec was also used for all the dataset versions. AraVec is “a pre-trained distributed word representation (word embedding) open-source project which aims to provide the Arabic NLP research community with free to use and powerful word embedding models” [3].

The approach was to see which word vectorizer performers better, then apply that to all the models.

Machine Learning Models

For modeling our data, we used 3 different machine learning algorithms which are (Decision Tress (DT), Naïve Bayes (NB) and Support Vector Machines (SVMs).

In the following paragraphs we will give a brief description for each Machine Learning algorithm

Decision Trees (DT)

Decision tree is a technique used for classification (supervised learning) purposes. It is a basic technique used in machine learning, data mining and business intelligence. DT is widely used since it does not require prior knowledge of the data or data domain and its processing is easier compared with other algorithms, it can also handle high dimensional data [4].

The way that decision tress work is by splitting or partitioning the data sets into subsets recursively based on a feature test [4]. The root of the tree contains the feature that has the higher value of the test amongst other features, the leaves of the tree (the last nodes) contain the classifications.

For our problem, a regular DT was constructed to classify each text to its category.

Naïve Bayes (NB)

Naïve Bayes is an algorithm used for supervised learning, it is based on the Bayes Theorem where the relationship between data classes is assumed to be independent and each class is given and equal weight as other classes [5]. Naïve bayes is one of the simplest machine learning algorithms, it is generally used for simple classification tasks and is considered a Lazy Learner.

Naïve Bayes works by making predictions for likelihoods of classes, meaning it calculates the

maximum probability of a data point to belong to a certain class and this is called the Naïve assumption [5].
For our project, NB was used to calculate the maximum probability of a certain text to belong to a certain category (topic).

The type of NB we used was the Gaussian Naïve Bayes classifier, it works similar to the original NB classifier, but it assumes normal distribution of the data [5].

Support Vector Machines (SVMs)

Support vector machine is an algorithm used for classification and prediction, that can handle both linear and nonlinear data, it is considered as a slow algorithm but is an extremely accurate algorithm. SVM has been applied to many applications, such as object recognition, handwritten digit recognition and speaker identification [6].

The goal of SVM is to find the best line or hyperplane which separates the data into classes. SVM works as follows, it uses a nonlinear mapping to renovate the data into a higher dimension, surrounded by this new dimension, it looks for the best line that separating the hyperplane, and it finds the line or hyperplane by using support vectors and margins [6].

In our case, SVMs will find the best line to separate the text based on their topic.

The type of SVMs used was Linear SVC (Support Vector Classifier).

For all the dataset versions, we split the data into a test size of 20% and a training size of 80% and the data was shuffled before the split to ensure that bias doesn't occur in our models and that the samples are generalized and random.

The general steps for modeling:

1. Running AraVec/Tf-idf on all the datasets.
2. Splitting the datasets for training and testing.
3. Specifying the model (NB, SVM, …etc.)
4. Fitting the model using the training datasets.
5. Predicting the topic using the testing datasets.
6. Storing the predictions in data dictionaries.

6. Using AraBERT for Model Training

Another approach used for modeling was the application of AraBERT. AraBERT is "an Arabic pretrained language model based on Google's BERT architecture, it was trained on ~70M sentences or ~23GB of Arabic text with ~3B words" [7].

AraBERT with Fast Bert was applied on the datasets that were preprocessed using MADAMIRA (Tokenization with Part-of-Speech datasets), with a total number of 5 epochs.

The table below capture the results of the Accuracy for all 5 versions of the datasets and 5 iterations.

	Epoch 1	**Epoch 2**	**Epoch 3**	**Epoch 4**	**Epoch 5**
Version 1	93	94	95	95	95
Version 2	94	94	95	95	95
Version 3	91	94	95	95	95
Version 4	89	94	93	94	94
Version 5	83	89	89	91	91

Table - 2: AraBERT with Fast BERT Accuracy Results

7. Evaluation Results

We computed the classification report for each model and for each dataset version. The tables contain the weighted average result in percentage for each evaluation measure.

Using AraVec Vectorizer

- Results For (Tokenization Only) Datasets using the 3 models

SVM MODEL

	Accuracy	**F1-score**	**Precision**	**Recall**
Version 1	95	95	95	95
Version 2	95	95	95	95
Version 3	95	95	95	95
Version 4	95	95	95	95

Version 5	95	95	95	95

NB MODEL

	Accuracy	**F1-score**	**Precision**	**Recall**
Version 1	88	88	89	88
Version 2	90	90	91	90
Version 3	91	91	91	91
Version 4	90	90	90	90
Version 5	86	87	87	86

DT MODEL

	Accuracy	**F1-score**	**Precision**	**Recall**
Version 1	77	76	77	77
Version 2	79	79	79	79
Version 3	78	78	78	78
Version 4	74	74	74	74
Version 5	67	67	68	67

Table - 3: Classification Report Results for Modeling (1)

- Results for (Tokenization + Part of Speech) Datasets using the 3 models

SVM MODEL

	Accuracy	**F1-score**	**Precision**	**Recall**
Version 1	96	96	96	96
Version 2	96	96	96	96
Version 3	96	96	96	96
Version 4	96	96	96	96

Version 5	96	96	96	96

NB MODEL

	Accuracy	**F1-score**	**Precision**	**Recall**
Version 1	89	89	89	89
Version 2	90	90	91	90
Version 3	90	91	91	90
Version 4	90	90	91	90
Version 5	86	86	87	86

DT MODEL

	Accuracy	**F1-score**	**Precision**	**Recall**
Version 1	79	79	79	79
Version 2	81	81	81	81
Version 3	77	77	78	77
Version 4	73	73	73	73
Version 5	68	68	69	68

Table - 4: Classification Report Results for Modeling (2)

Using TF-IDF Vectorizer

- Results For (Tokenization Only) Datasets using the 3 models

SVM MODEL

	Accuracy	**F1-score**	**Precision**	**Recall**
Version 1	97	97	97	97
Version 2	97	97	97	97
Version 3	97	97	97	97

Version 4	97	97	97	97
Version 5	97	97	97	97

NB MODEL

	Accuracy	**F1-score**	**Precision**	**Recall**
Version 1	91	91	91	91
Version 2	88	88	88	88
Version 3	86	86	87	86
Version 4	85	86	86	85
Version 5	75	75	77	75

DT MODEL

	Accuracy	**F1-score**	**Precision**	**Recall**
Version 1	78	78	78	78
Version 2	75	75	75	75
Version 3	77	77	78	77
Version 4	75	75	76	75
Version 5	76	76	76	76

Table - 5: Classification Report Results for Modeling (3)

- Results for (Tokenization + Part of Speech) Datasets using the 3 models

SVM MODEL

	Accuracy	**F1-score**	**Precision**	**Recall**
Version 1	97	97	97	97
Version 2	97	97	97	97
Version 3	97	97	97	97

Version 4	97	97	97	97
Version 5	97	97	97	97

NB MODEL

	Accuracy	**F1-score**	**Precision**	**Recall**
Version 1	91	91	91	91
Version 2	88	88	88	88
Version 3	86	86	87	86
Version 4	85	86	86	85
Version 5	75	75	77	75

DT MODEL

	Accuracy	**F1-score**	**Precision**	**Recall**
Version 1	80	80	80	80
Version 2	77	77	77	77
Version 3	76	76	76	76
Version 4	74	74	76	74
Version 5	76	76	76	76

Table - 6: Classification Report Results for Modeling (4)

Evaluation Conclusions

To summarize, four performance measures were computed: Accuracy which is the percentage of items that are correct, F1-score which is a combined measure that assesses the precision/ recall tradeoff, Precision which is the percentage of selected items thar are correct and Recall which is the percentage of correct items that are selected.

As mentioned, we tried 3 models SVM, NB and DT, using 2 types of word embedding vectors (Tf-idf and AraVec) and 2 types of datasets each with 5 versions.

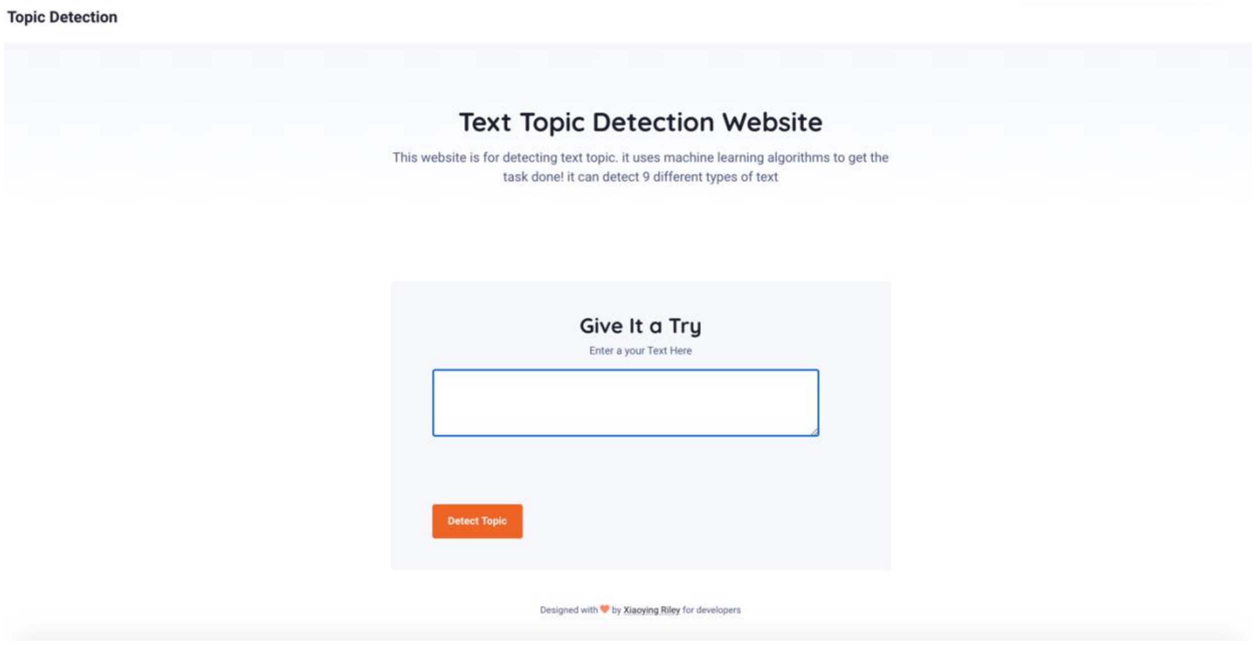
The best performance from the models was for the SVM model which had the accuracy of 97% for all versions using (TOK + POS) datasets, the worst performance was for the DT model.

8. Deployment

We used Flask Library from python to deploy our work, we used one of the best results for the SVM models to deploy along with the best result for the dataset version (version1).

Looking at the results, Tf-idf vectorizer performed better than AraVec vectorizer, but since the application was build based on Arabic datasets, we felt it’s suitable to use AraVec rather than Tf-idf. This was also done to enrich the Arabic Natural Language Processing community.

We deployed the model as an HTML simple page, where any user can input an Arabic Text and the application will detect the topic of that text.



Topic Detection

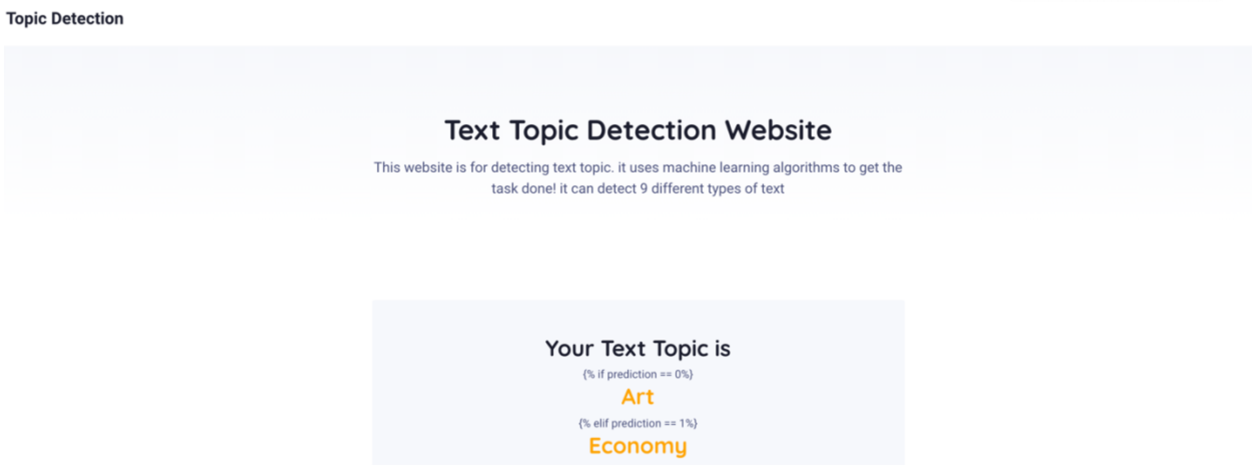
Text Topic Detection Website

This website is for detecting text topic. it uses machine learning algorithms to get the task done! it can detect 9 different types of text

Give It a Try

Enter a your Text Here

Designed with ♥ by Xiaoying Riley for developers



Topic Detection

Text Topic Detection Website

This website is for detecting text topic. it uses machine learning algorithms to get the task done! it can detect 9 different types of text

Your Text Topic is

{% if prediction == 0%}

Art

{% elif prediction == 1%}

Economy

9. Limitations and Future Work

One of the limitations we faced was the lack of resources regarding the use of MADAMIRA to tokenize the text. The examples found were limited and didn't cover different types of raw inputs. And as for the future work, the topic feature had 9 classes of categories, additional categories can be added such as: History, Environment, … etc. Also, other features can be generated, for example we can make a feature that has the main category (i.e. Education can include (Literature, Art, Technology)). This can help us to produce a multi-class classification model.

10. Conclusion

We conclude that it is possible to preform Topic Detection task to classify texts into their appropriate topic using various modeling algorithms, and these algorithms can perform the task with high accuracies and efficiency. These models can then be used by various organizations that need this task to improve their work.

References

1. https://monkeylearn.com/topic-analysis/
2. A. Arfath et al., "MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic", pp. 1-8. [Accessed 5 November 2020].
3. https://github.com/bakrianoo/aravec
4. https://www.geeksforgeeks.org/decision-tree/
5. https://www.geeksforgeeks.org/naive-bayes-classifiers/
6. D. Vijayarani and M. Dhayanand, "KIDNEY DISEASE PREDICTION USING SVM AND ANN ALGORITHMS", International Journal of Computing and Business Research (IJCBR), vol. 6, no. 2, 2015. [Accessed 5 November 2020].
7. https://github.com/aub-mind/arabert