

Machine Learning Approaches for Arabic Dialect Identification using Monodialectal Language Models

By

Reem AlYami

A Thesis Presented to

THE DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of
Requirements for the Degree of

MASTER OF SCIENCE

In

Computer Science

March 2022

MACHINE LEARNING APPROACHES FOR
ARABIC DIALECT IDENTIFICATION USING
MONODIALECTAL LANGUAGE MODELS

BY

REEM ALYAMI

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

In Partial Fulfillment of the Requirements
for the Degree of

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

KING FAHD UNIVERSITY
OF PETROLEUM & MINERALS

Dhahran, Saudi Arabia

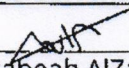
MARCH, 2022

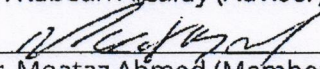
KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

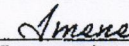
DEANSHIP OF GRADUATE STUDIES

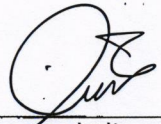
This thesis, written by **REEM ALYAMI** under the direction of her thesis adviser and approved by her thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER SCIENCE**.

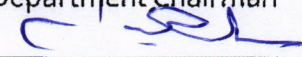
Thesis Committee


Dr. Rabeah AlZaidy (Adviser)


Dr. Moataz Ahmed (Member)


Dr. Imane Boudelloua (Member)


Dr. Hamoud Aljamaan
Department Chairman


Dr. Suliman AlHomidan
Dean of Graduate Studies

Date



©Reem AlYami
2022

To the memory of a great man that I was fortunate to have as my
Father.

ACKNOWLEDGMENTS

All the praise and thanks be to Allah who has guided me to this. I never could I have found guidance were it not that Allah has guided me. I am grateful to many people who were there for me during my thesis journey. I would like to thank my mom and siblings for their prayers, encouragement, and constant support. I also want to thank my advisor for her patience, knowledge, and guidance during my thesis work. I want to thank my friends; I am deeply fortunate to have you as my friends. Also, I would like to thank my KFUPM friends for our group meetings, discussions, and brainstorming. Thank you for making this journey memorable.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	vi
LIST OF TABLES	xi
LIST OF FIGURES	xiii
ABSTRACT (ENGLISH)	xviii
ABSTRACT (ARABIC)	xxi
CHAPTER 1 INTRODUCTION	1
1.1 Research Motivation	1
1.2 Problem Statement	2
1.3 Research Objectives and Questions	3
1.4 Research Contribution	4
1.5 Thesis Organization	5
CHAPTER 2 BACKGROUND	7
2.1 Arabic Dialect Identification	7
2.1.1 Arabic Language Varieties	7
2.1.2 Arabic Dialect Varieties	11
2.1.3 Automatic Arabic Dialect Identification	12
2.2 Language Modeling	18
2.2.1 Statistical Language Models	19
2.2.2 Distribution Representation Language Models	21

2.3	Contextual Representation Language Models	24
2.3.1	Encoder-Decoder Networks	25
2.3.2	Attention	27
2.3.3	Self-Attention and Transformer Networks	29
2.3.4	Bidirectional Encoder Representation from Transformers	37
CHAPTER 3	LITERATURE REVIEW	45
3.1	Arabic Dialect Resources	45
3.1.1	Arabic Corpora Data Sources	46
3.1.2	Arabic Corpora	47
3.2	Arabic Dialect Identification	49
3.2.1	Dialectal Granularity Level	49
3.2.2	Features	50
3.2.3	Identification Method	52
3.2.4	Evaluation Metrics	53
3.3	Arabic Neural Language Modeling	54
3.3.1	Neural Language Models	55
3.3.2	Non-Contextualized Representation for Arabic	55
3.3.3	Contextualized Representation for Arabic	56
3.4	Gap Analysis	56
3.4.1	Arabic Dialect Resources	57
3.4.2	Arabic Dialect Identification	58
CHAPTER 4	BUILDING ARABIC DIALECT RESOURCES	61
4.1	Motivation	61
4.2	Arabic Dialect Resources	62
4.3	Dialectal Short-Text Corpus Collection	65
4.3.1	Twitter Data Collection	66
4.3.2	Geographical Coordinate Data Collection	68
4.3.3	Twitter Trends Data Collection	68
4.3.4	Data Evaluation	71

4.4	Dialect Dictionary Collection	72
4.4.1	Crowd-sourced Dialect Dictionary	72
4.4.2	Curated Dialect Dictionary	73
4.4.3	Arabic General Purpose Lexicon (AGPL)	73
4.5	Data Preprocessing	74
4.5.1	Dialect Short-Text Corpus	75
4.6	Building a Manually Annotated Corpus	77
4.6.1	Manual Annotation	78
4.6.2	Negative Sample Generation	79
4.7	Building Automatically Annotated Corpus	80
4.7.1	Building a Dialectal Dictionary	80
4.7.2	Automatic Annotation	82
4.7.3	Negative Sample Generation	86
CHAPTER 5 ARABIC CONTEXTUAL REPRESENTATION		88
5.1	Motivation	88
5.2	Transformer-Based Language Models	89
5.3	Arabic RoBERTa (AraRoBERTa)	93
5.3.1	Training Data	94
5.3.2	Tokenizers for Arabic Dialects	94
5.3.3	AraRoBERTa-XX Models	95
5.3.4	Training Optimization	97
5.3.5	Results	98
CHAPTER 6 ARABIC DIALECT IDENTIFICATION		103
6.1	Supervised Learning Approach	103
6.1.1	Classification Model	104
6.1.2	Evaluation	104
6.2	Weakly-supervised Learning Approach	106
6.2.1	Classification Models	107
6.2.2	Evaluation	109

6.3	Semi-supervised Learning Approach	114
6.3.1	Classification Models	115
6.3.2	Evaluation	118
CHAPTER 7	ADI EVALUATION	122
7.1	Baseline Models	122
7.2	Experimental Results	124
7.3	Discussion	125
CHAPTER 8	CONCLUSION AND FUTURE WORK	130
8.1	Conclusion	130
8.2	Future Work	131
8.2.1	Arabic Dialect Resources	131
8.2.2	Arabic Dialect Annotations	132
8.2.3	Arabic Dialect Similarities	132
REFERENCES		133
VITAE		159

LIST OF TABLES

Table 3.1	Summary on the recent work on Arabic Corpora.	49
Table 3.2	Summary of the recent work on Arabic dialect identification.	54
Table 4.1	Country coordinates.	69
Table 4.2	Country WOEID [1].	70
Table 4.3	Average daily statistics for running the proposed Twitter pipeline. . . .	71
Table 4.4	Data validation for the collected data from the proposed pipeline. . . .	72
Table 4.5	Crowd-sourced dialectal dictionary.	73
Table 4.6	Curated dialectal dictionary.	73
Table 4.7	Twitter dataset size and the effect of cleaning on the tweets	75
Table 4.8	Manually Annotated Data.	79
Table 4.9	Negative Sample Generation for Each Dialect.	80
Table 4.10	Building Dialect Dictionary Process	82
Table 4.11	Automatically Annotated Data Size.	85
Table 4.12	The Evaluation of the Weak Annotation Approach.	86
Table 4.13	Negative Sample Generation for each Dialect.	87
Table 5.1	AraRoBERTa pre-training data.	95
Table 5.2	AraRoBERTa Pre-training Configuration	96
Table 5.3	AraRoBERTa hyperparameters for Adam optimization algorithm. . . .	97
Table 6.1	AraRoBERTa models performance against traditional ML model. *	
	denotes the different AraRoBERTa variations corresponding to each	
	dialect.	106
Table 6.2	The training data split for WSL-I.	110

Table 6.3	The training data split for WSL-II.	110
Table 6.4	Weak-supervised dialect classification results. Best results are in bold and blue cell. The green cell indicate the best results between the weak classifiers.	112
Table 6.5	The WSL-I performance on the validation and test sets.	112
Table 6.6	The predictions of WSL-I AraRoBERTa Classifier. The Highest Incorrect Predictions are bold.	112
Table 6.7	The performance change between WSL-I and WSL-II models.	114
Table 6.8	The produced data during the SSL-II training.	120
Table 6.9	Semi-supervised dialect classification results. Best results are in bold and blue cells. Green cells indicate the best results between the SSL classifiers.	121
Table 7.1	Configurations of existing models and AraRoBERTa models. WP is WordPiece and SP is SentencePiece	123
Table 7.2	The performance of the AraRoBERTa models against other contextual models. * denotes the different AraRoBERTa models corresponding to each dialect.	125

LIST OF FIGURES

Figure 2.1	Arabic dialects categorized based on the geographical location.	8
Figure 2.2	Different regions in the Arab world.	9
Figure 2.3	Arabic letters that are pronounced differently among dialects.	11
Figure 2.4	The Arabic word corresponding to the word "I Write" written in two dialectal formats shown below each word.	12
Figure 2.5	Text classification process.	13
Figure 2.6	Word2Vec: CBOW architecture.	23
Figure 2.7	Word2Vec: Skip-gram architecture.	24
Figure 2.8	The training process for a neural language modeling approach for ma- chine translation. The pair bitexts are concatenated and used to train a language model.	26
Figure 2.9	Simple RNN-based encoder-decoder architecture. The RNN encoder provides the decoder with the final hidden state as the context.	27
Figure 2.10	The encoder-decoder in a Transformer.	30
Figure 2.11	Transformer architecture.	31
Figure 2.12	Self-Attention Example [2].	35
Figure 2.13	BERT provides the embedding representation of an example sentence.	39
Figure 2.14	BERT learning settings (1) Learning: self-supervised trained on large textual datasets. (2) Transfer learning: supervised learning on a spe- cific downstream task with an annotated dataset.	40
Figure 2.15	BERT input embedding representation.	42
Figure 2.16	Illustrations of transfer learning in BERT on the text classification task.	44
Figure 3.1	Data and annotation gap.	58

Figure 3.2	Representation and dialect identification gap.	60
Figure 4.1	Arabic dialect resources.	63
Figure 4.2	The pipeline of building an annotated Arabic dialect corpus.	64
Figure 4.3	The developed Twitter data collection pipeline	67
Figure 4.4	Specifying a Centering Geographical Point in Saudi Arabia Using Free map Online Tool [3].	69
Figure 4.5	Preprocessing Steps on a Sample Tweet.	77
Figure 4.6	Supervised Learning Annotation Process.	79
Figure 4.7	The Process of Building Arabic Dialectal Dictionaries.	81
Figure 4.8	Tweets Automatic Annotation Process.	84
Figure 4.9	Sample Tweet that is Automatically Annotated.	84
Figure 4.10	Samples from the Weak Positive Data for Different Dialects.	87
Figure 5.1	BPE text processing example.	92
Figure 5.2	The process of developing AraRoBERTa.	94
Figure 5.3	Training loss of AraRoBERTa-ALL: 340,000 steps equals 51 hours of split training on 2,2,2,2,2 epochs.	99
Figure 5.4	Training loss of AraRoBERTa-SA: 161,500 steps equals 25 hours of split training on 3,4,3 epochs.	99
Figure 5.5	Training loss of AraRoBERTa-EG: 133,500 steps equals 22 hours of split training on 3,4,3 epochs.	100
Figure 5.6	Training loss of AraRoBERTa-KU: 31,000 steps equals 6 hours of split training on 5,5 epochs.	100
Figure 5.7	Training loss of AraRoBERTa-OM: 16,000 steps equals 7 hours of split training on 5,5 epochs.	101
Figure 5.8	Training loss of AraRoBERTa-JO: 9,000 steps equals 4 hours of con- tinuous training on 5,5 epochs.	101
Figure 5.9	Training loss of AraRoBERTa-LB: 16,000 steps equals 3 hours of con- tinuous training on 10 epochs.	102

Figure 5.10	Training loss of AraRoBERTa-DZ: 6,500 steps equals 2 hours of continuous training on 7,3 epochs.	102
Figure 6.1	Supervised ADI on hand-labeled data.	105
Figure 6.2	The WSL-I model for ADI task.	108
Figure 6.3	The WSL-II model for ADI task.	109
Figure 6.4	Confusion matrix for AraRoBERTa-KU in the supervised and weak supervised setup.	113
Figure 6.5	SSL-I model for ADI task.	117
Figure 6.6	The SSL-II model for ADI task.	119
Figure 7.1	Confusion matrix for AraRoBERTa-OM.	126
Figure 7.2	AraRoBERTa-SA confusion matrix and a misclassified sample.	127
Figure 7.3	AraRoBERTa-LB confusion matrix and a misclassified sample.	127
Figure 7.4	AraRoBERTa-DZ confusion matrix, the misclassification in false positive and negative predictions, and the true positive and negative are comparable.	128
Figure 7.5	AraRoBERTa-JO confusion matrix, the misclassification is in false positive and negative, and true positive and negative sample predictions are comparable.	128
Figure 7.6	AraRoBERTa-EG confusion matrix, the misclassification is in false positive and negative, and true positive and negative sample predictions are comparable.	129
Figure 7.7	Multi-dialect identification confusion matrix.	129

THESIS ABSTRACT

NAME: Reem AlYami

TITLE OF STUDY: Machine Learning Approaches for Arabic Dialect Identification using Monodialectal Language Models

MAJOR FIELD: Computer Science

DATE OF DEGREE: March, 2022

The rapid advancement of applications built on Arabic language processing tasks requires addressing performance challenges that are specific to the Arabic language. Arabic belongs to the group of *diglossic* languages, where different variations of the language are spoken among speakers. Identifying Arabic dialects is essential to provide better technologies and services. Although many pioneering efforts have focused on collecting and annotating the data, the openly available data remains lacking in quantity and quality. For example, Arabic datasets suffer from issues such as investigating the dialect in a specific domain which limits the text diversity. Additionally, the Arabic dialect granularity level poses a challenge when investigating a specific dialect.

This thesis aims to identify Arabic dialects by providing contextual feature representation for Arabic dialects and using different learning paradigms to classify dialects. This work develops a data-collection pipeline that is language- and task-agnostic for collecting Arabic dialects. This method collects data at the country level to provide a diverse sample for each dialect. This data is used for developing dialect-specific contextual language models (AraRoBERTa) to classify Arabic dialects. Additionally, an automatic annotation method for Arabic dialects is proposed to identify dialects. Also, two learning paradigms are applied, Weak-Supervised Learning (WSL) and Semi-supervised Learning (SSL), to identify Arabic dialects and thoroughly evaluate them. WSL has two settings: learning from automatically labeled data (WSL-I) and learning from weakly labeled data after reducing the noise (WSL-II). Additionally, SSL is used to learn from the available data to enhance the performance of the models in two settings: learning from a group of unlabeled dialects (SSL-I) and learning from a group of pseudo-labeled dialects (SSL-II).

Different Arabic resources are built for seven Arabic country-level dialects: Saudi Arabia, Kuwait, Oman, Lebanon, Jordan, Egypt, and Algeria. These resources are used to develop variations of the AraRoBERTa language models. The results show that developing dialect-specific AraRoBERTa yields competitive results. The best variation AraRoBERTa-EG reached 93% F-1, improving over supervised machine learning baselines and other contextual models 1% to 6%. Weak-supervision for Arabic dialects is not ideal. However, it is a promising way to train models on larger data. Finally, the semi-supervised approaches yielded the best performance in three dialects, especially for AraRoBERTa variations trained on smaller data sizes. An im-

provement of around 8% and 3% over supervised are reached for AraRoBERTa-OM and AraRoBERTa-LB, respectively.

ملخص الرسالة

الاسم: ريم اليامي

عنوان الدراسة: طرق التعلم الآلي لتحديد اللهجات العربية باستخدام نماذج سياقية أحادية اللغة

التخصص: علوم الحاسب

تاريخ الدرجة العلمية: مارس - ٢٠٢٢

أصبحت تحديات الأداء التي تحتاج إلى معالجة باللغة العربية أكثر إلحاحًا مع التقدم المستمر في مهام معالجة اللغة. تنتمي اللغة العربية إلى مجموعة اللغات ثنائية اللغة ، حيث يتم التحدث بلهجات مختلفة بين المتحدثين. يعد التعرف على اللهجات العربية أمرًا ضروريًا لتوفير تقنيات وخدمات أفضل. على الرغم من أن العديد من الجهود الرائدة عملت على جمع البيانات وقليل منها على توسيم البيانات ومع ذلك ، تظل البيانات المتاحة ضئيلة وتعاني من مشكلات مثل تجميع لبيانات اللهجة في مجال معين مما يحد من تنوع النص. بالإضافة إلى ذلك ، يشكل مستوى التعرف على اللهجة العربية تحديًا عند تصنيف اللهجة.

وبالتالي ، تهدف هذه الأطروحة إلى التعرف على اللهجات العربية من خلال توفير تمثيل سياقي لسمات اللهجات العربية واستخدام نماذج تعليمية مختلفة لتصنيف اللهجات. تطور هذه الدراسة آلية لجمع البيانات غير معتمد على اللغة أو المهمة في جمع اللهجات العربية. تجمع البيانات على مستوى الدولة لتوفير عينة متنوعة لكل لهجة لتطوير نماذج لغة سياقية خاصة باللهجة (AraRoBERTa) لتصنيف اللهجات العربية. بالإضافة إلى ذلك ، تم اقتراح طريقة تعرف آلية للهجات العربية لتحديد اللهجات. حيث يتم تقييم نموذجين للتعلم: التعلم الخاضع للإشراف الضعيف (WSL) والتعلم شبه الخاضع للإشراف (SSL) لتحديد اللهجات العربية. يعمل WSL على إعدادين: التعلم من البيانات الموسومة آلياً (WSL-I) والتعلم من البيانات الموسومة آلياً بعد تقليل العينات الغير مفيدة. (WSL-II) بالإضافة إلى ذلك، يتم استخدام SSL للتعلم من البيانات المتاحة لتحسين

أداء النماذج في إعدادين: التعلم من مجموعة من بيانات اللهجات الغير موسمة (SSL-I) والتعلم من مجموعة من بيانات اللهجات ذات العلامات التقرية (SSL-). (SSL-)

نتيجة لذلك، تم إنشاء مصادر عربية مختلفة لسبع لهجات عربية: السعودية والكويتية والعمانية واللبنانية والأردنية والمصرية والجزائرية. تُستخدم هذه الموارد لتطوير أشكال مختلفة من AraRoBERTa. تظهر النتائج أن تطوير نموذج اللغة السياقي AraRoBERTa الخاصة باللهجة يؤدي إلى نتائج تنافسية. وصلت أفضل نسخة من AraRoBERTa-EGY إلى 93 ٪، وأدت بشكل أفضل من نماذج ذكاء الآلة التقليدية 1٪ إلى 6 ٪. كذلك التعلم من البيانات الموسمة آلياً يعتبر طريقة واعدة لتدريب النماذج على بيانات أكبر. أخيراً، أسفرت النماذج شبه الخاضعة للإشراف عن أفضل أداء في ثلاث لهجات، خاصة بالنسبة إلى أشكال AraRoBERTa المدربة على أحجام بيانات أصغر. تم الوصول إلى تحسن من 3٪ إلى 8٪ أكثر من الإشراف بالنسبة لـ AraRoBERTa-OM و AraRoBERTa-LB .

CHAPTER 1

INTRODUCTION

This chapter presents the motivation for the thesis, the research problem, the objectives and the questions raised, the contribution of this work, and finally, the thesis organization.

1.1 Research Motivation

Arabic is the *5th* most spoken language in the world with over 422 million speakers [4]. Languages with this scale of users, such as English, are heavily investigated in the literature [5]. This reflects the availability of the resources for these languages, including datasets and Natural Language Processing (NLP) tools that enable more research and widespread adoption [5]. However, the case for Arabic is different as it is still considered a low-resource language in the literature of NLP [6]. A large amount of Arabic work focuses on Modern Standard Arabic (MSA), which is one of the Arabic varieties that is used in formal writing and taught in schools to Arabic speakers. However, informal communication among Arabic speakers occurs through

informal local diglossic dialects [7].

Diglossia refers to having multiple varieties of a single language among the language speakers. The Arabic diglossia poses a need to identify the dialect before performing different NLP tasks such as tokenizing, parsing, and other downstream tasks, for instance, semantic inferences [6], [8]. Large amounts of text found in online services and platforms are written in the local vernacular, which requires dialect identification to enhance system performance [9]. Additionally, Arabic dialect identification helps in identifying user demographic in contexts related to customer reports, health forums, tourism and entertainment reviews, and many others, which ultimately leads to improved services for users from each demographic.

1.2 Problem Statement

The Arabic dialect identification (ADI) task is essential to provide improved technologies and services. However, the available datasets for Arabic dialects remain minimal compared to other rich resource languages such as English [10]. Although several pioneering efforts worked on addressing this issue by collecting data, the collected data is typically developed for a specific task such as sentiment analysis or a specific domain such as travel which limits the text diversity in representing the dialect [11], [12], [13].

Data annotation is an essential step in the ADI task, where the efforts focus mainly on manual annotation requiring the allocation of resources, which can be costly [14]. Although few works utilize the metadata as an automatic annotation method, the location is not always available in the collected text [15], [16]. Dialect feature repre-

sensation is another essential step in the ADI task. Although many efforts have worked on utilizing statistical models for representing dialects, no performance change is encountered compared to multilingual contextual features or monolingual MSA features that yielded promising results. The Arabic dialect’s contextual features remain unexplored. Additionally, the granularity level poses a challenge in Arabic dialects where classifying the dialects in the region is linguistically inaccurate, as indicated by many studies, and the city-level yields low-performance [6], [17], [18]. Hence, there is a need to propose an ADI model utilizing the Arabic dialect contextual features that capture features based on Arabic dialect corpora with sufficient variations that reflect the dialect’s nature. Also, to have an effective Arabic dialect classification, there is a need to explore different supervised learning paradigms to exploit manual annotation and explore automatic annotation on data for Arabic dialects.

1.3 Research Objectives and Questions

The main objective of this thesis is to answer the following research questions (RQs):

1. **RQ1:** How can we effectively build datasets that can be used for the ADI problem?
2. **RQ2:** What method can be used to annotate a large volume of Arabic dialect data?
3. **RQ3:** To what extent can using contextualized learning improve the performance of Arabic dialect identification?

Following these RQs, the research objectives mapped to the corresponding RQs are as follows:

1. Investigate a data collection approach for collecting different Arabic dialects.
(Maps to RQ1)
2. Develop an automatic annotation method to minimize the annotation overhead in the context of Arabic dialects. (Maps to RQ2)
3. Develop a contextual representation language model to represent the dialect features for each Arabic dialect included in this study. (Maps to RQ3)
4. Develop Arabic dialect identification models using different supervised learning paradigms. (Maps to RQ3)

1.4 Research Contribution

This thesis work focuses on Arabic dialect identification and Arabic dialectal resources. In order to address the shortcomings of using statistical representation for the ADI task, this work develops a dialect-specific contextual representation language model for each investigated dialect. In addition, a multi-dialect representation of all the investigated dialects is developed for use in general Arabic NLP. As a result of the shortage of publicly available data for Arabic dialects, this work proposes a novel data collection pipeline from Twitter that is language- and task-agnostic in order to collect data from seven Arabic countries. Three learning approaches are proposed and evaluated. The first approach is a supervised learning ADI using contextual representation. This

approach relies on constructing manually annotated data to address the shortage in the annotated data. The second approach is a weakly-supervised learning model that requires automatically annotated data. This work proposes an automatic annotation method and develops dialect-specific dictionaries for each dialect to address the shortage of annotated data. Also, in this approach, we propose two methods that can be utilized when performing the ADI task. The final approach, semi-supervised learning, uses both labeled and unlabeled data while maintaining model generalization criteria. Also, in this approach, we propose two novel methods that can be utilized when performing the ADI task. Finally, the performance of Arabic dialect identification in different learning settings is extensively evaluated.

1.5 Thesis Organization

The remaining chapters of the thesis are organized as follows.

- Chapter 2 provides a background on Arabic and its dialect and the different techniques used for feature representation.
- Chapter 3 summarizes the surveyed literature related to Arabic resources and Arabic dialect identification and provides the identified gap analysis.
- Chapter 4 describes the proposed data collection pipeline, the process of collecting different dictionaries, the data preprocessing, and building the manual and automatically annotated corpora.
- Chapter 5 describes the method for representing the features of Arabic dialects.

- Chapter 6 describes the Arabic dialect identification task, the adopted approaches to identify Arabic dialects, and the obtained results.
- Chapter 7 discusses the adopted approaches to identify Arabic dialects and the results.
- Chapter 8 concludes the findings and provides the limitations and future directions.

CHAPTER 2

BACKGROUND

This chapter provides a background on Arabic Dialect Identification (ADI), traditional feature representation as statistical language models, and more recent contextual language models.

2.1 Arabic Dialect Identification

This section provides an overview of the Arabic language varieties, Arabic dialect varieties, and the automatic ADI task.

2.1.1 Arabic Language Varieties

Arabic as a language is a diglossic language with many varieties that co-exist concurrently [19]. These varieties include MSA and Dialectal Arabic (DA). MSA is the formal Arabic written in books, news, and formal communication, and shared among Arabic speakers [4]. DA is the informal variety of Arabic used in daily interactions among Arabic speakers [4]. In recent years, a growing interest in DA has appeared due

to the use of DA in written communication, such as social networks, blogs, forums, and reader commentary [20].

In terms of writing systems, i.e., characters used to inscribe the language, the majority of DA uses the Arabic script, that is, around 22 dialects out of 36 DA varieties [20]. Arabic script is the most widely used after the Latin script. There is no lower or upper case for the Arabic letters; however, some letters can differ in writing according to their position in the word [20]. All Arabic varieties share the same alphabet that consists of 28 letters and three optional diacritical marks: vowel, nunation, and shaddah [20]. Grouping of DA can be done in a variety of ways [20]; where the predominant one is to categorize DA through the geographical location; the number of Arabic dialects is 36 as Figure 2.1 shows.

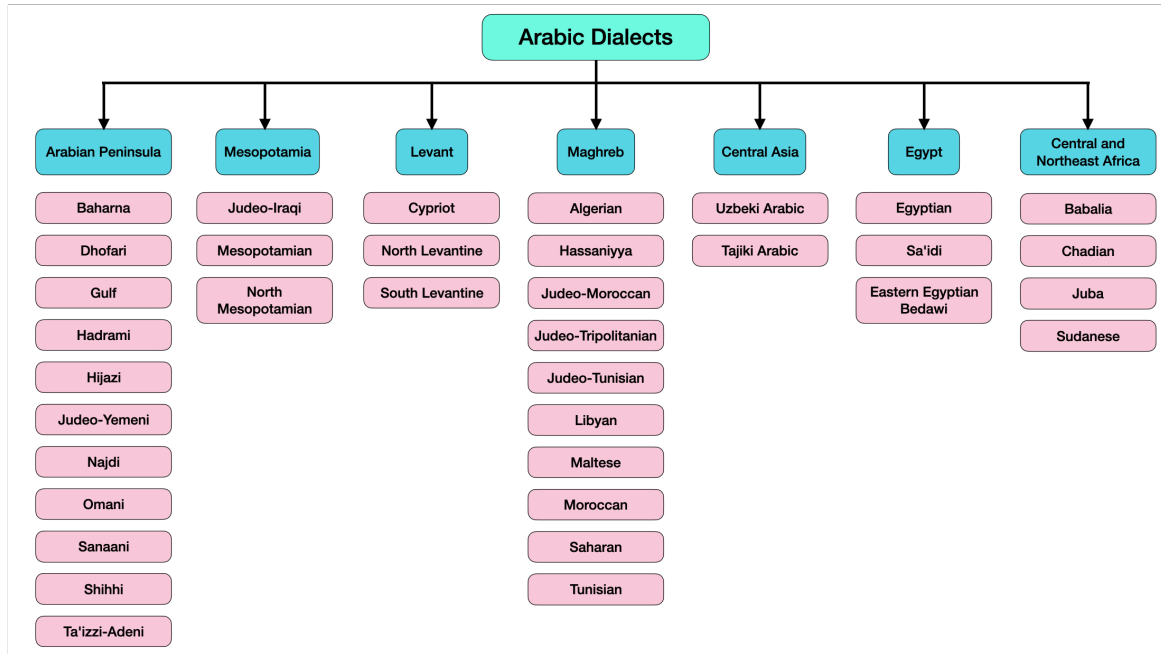


Figure 2.1: Arabic dialects categorized based on the geographical location.

Arabic dialects can be further divided into several dialectal varieties based on the granularity level, namely, region, country, and city. The regional level represents

different regions in the Arab world consisting of a set of countries as shown in Figure

2.2. These regions are as follows [17], [6]:

1. **Gulf:** includes six countries of the Arabian Peninsula: Saudi Arabia, Kuwait, United Arab Emirates, Bahrain, Oman, and Qatar.
2. **Yemen:** is a country; however it is categorized as a regional dialect.
3. **Iraq:** is a country as well; however, it is categorized as a regional dialect.
4. **Levantine region:** includes four countries: Jordan, Lebanon, Palestine, and Syria.
5. **Nile Basin:** includes two countries, Egypt and Sudan.
6. **North Africa:** includes five countries: Algeria, Tunis, Morocco, Libya, and Mauritania.

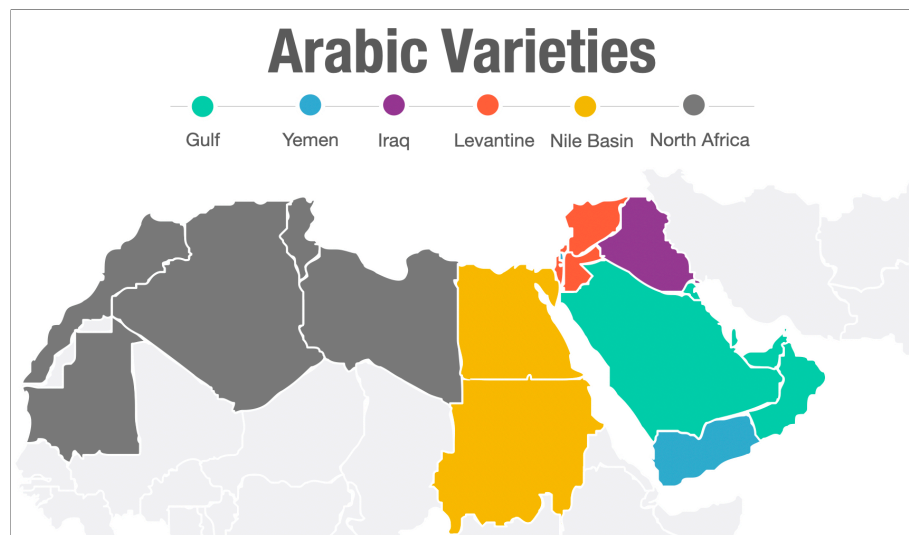


Figure 2.2: Different regions in the Arab world.

Arabic Varieties Linguistic Differences

The differences between MSA and DA originate from different linguistic perspectives, that is, the phonological, morphological, and syntactic or lexical level [20], [21], [4], [22]. A brief summary of some differences in phonological and morphological perspectives that are most relevant to this study is provided below.

Phonological Differences: phonology is a field in linguistics that studies the sounds in a language or dialect. Here we list some examples of the phonological differences between MSA and DA [20], [21]:

- In MSA, the interdental fricatives *th* is replaced by the constant letter *or*, as what is known in phonology as alveolar stop, the letter *t* in some dialects such as Syrian Arabic.
- In MSA, the words containing a glottal stop in the middle are not present in DA. However, some Arabic dialects use glottal stops to replace another phoneme.

Morphological Differences: morphology is a field in linguistics that studies the structure of the word in a language or a dialect. Here we list some examples of the morphological differences between MSA and DA [20], [4]:

- In MSA, there is a dual form when addressing a pair in speech. However, dialects lack the dual form and usually opt for the plural form in verb conjugation and pronouns.
- Several dialects do not have gender distinction between male and female. However, in MSA, gender is present morphologically in all word classes except in a

few cases such as pronouns.

- MSA has a complex case ending system where words can have diacritical marks attached to the word's ending to reflect a grammatical function. Arabic dialects do not possess this phenomenon.

2.1.2 Arabic Dialect Varieties

Arabic dialects differ on all linguistic levels, that is, phonological, morphological, and syntactical levels [20]. These differences are not only on the regional dialect level but also on the country level [20]. Arabic dialects differ from a phonological perspective, mainly in the pronunciation of six letters shown in Figure 2.3. These letters are replaced with another letter when pronounced or pronounced in different ways [20].

ظ	ض	ذ	ث	ق	ج
---	---	---	---	---	---

Figure 2.3: Arabic letters that are pronounced differently among dialects.

Arabic dialects have morphological differences, and these variations can be seen between eastern and western Arabic dialects in the imperfect verb in the prefix of the singular person [20]. For example, some verbs in the Arabian Peninsula dialects are written as 'I write', and the other dialect writes it in a manner that corresponds to 'We write' although both of these words refer to 'I write' as Figure 2.4 shows [20].

Other syntactical differences between DA varieties are present as well. For example, in Syrian Arabic, the questioning words such as 'what', 'where', and 'when' are present at the end of the question, which is opposite to other dialects [20], [21].

أكتب	نكتب
Arabian Peninsula	Moroccan

Figure 2.4: The Arabic word corresponding to the word "I Write" written in two dialectal formats shown below each word.

Another notable difference is the use of demonstrative articles after the nominal word in Egyptian Arabic, although other Arabic dialects use it before and after the nominal word [23]. Additionally, the meaning of vocabulary between dialects can differ greatly, a certain word can have one meaning in one dialect, and a different meaning in another dialect [20], [23].

2.1.3 Automatic Arabic Dialect Identification

The ADI in the text is the process of identifying the dialect of a sequence of words or a document [24]. Formally, it is the process of developing a classifier that can recognize a specific text T as belonging to a particular dialect D based on the provided annotation tags attached to the text [24], [25]. Arabic dialect identification is considered a difficult language identification task since Arabic dialects have the same alphabet and have some shared vocabulary [4], [26]. However, dialects have unique characteristics that can aid in the dialect classification task [4], [26]. These characteristics are considered challenges, as discussed in the following section, since they require collecting data and building tools for each dialect. At the same time, it is what makes these dialects classifiable. Text classification in NLP can be rule-based or use machine learning models [27]. In ADI, the task is mainly a supervised learning task where the learning model is given a text with a predefined class associated with it [28]. Then the model

learns from and applies it to unseen text to assign classes for it [28].

The ADI task can be formalised as follows given a set of short text, $D = \{(t_1, y_1), (t_2, y_2), \dots, (t_n, y_n)\}$ where T denote the short text instances, n denotes the number of instances and the label is denoted by $Y = \{P, N\}$ where P represents a specific Arabic dialect, and N represents the negative samples that do not belong to the dialect. The model performs binary classification to assign each t_i a y_j label. Dialect classification is similar to other classification tasks. It includes several steps: collecting text, preprocessing raw text, learning from text, and evaluation as Figure 2.5 illustrates [28].

The preprocessing step is essential when working on textual data. This step includes several steps that studies empirically evaluate, and it can consume up to 80% of the research effort [28]. These steps include text cleaning and representation. The former includes the removal of stopwords, punctuation marks, and irrelevant symbols in text [28]. The feature representation is a crucial step when classifying text [28]. In order to build an effective learning model, the text needs to be transformed into an internal representation, such as feature vectors that allow models to learn from it [29].

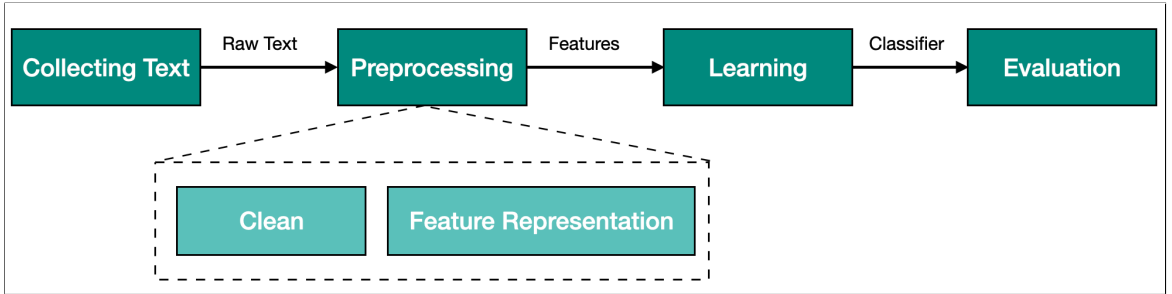


Figure 2.5: Text classification process.

The representation of text is critical in determining the usefulness of the extracted

features from the raw text in the classification task [29]. When the representation extracts more meaningful information from the text, the performance of the classification or prediction task can be increased [29]. Making data representation a crucial step in the classification process [29]. After the preprocessing step, the data is ready to be passed to the learning algorithm to fit a classification model [28], [29]. Then the learned algorithm is evaluated using the defined evaluation metrics selected by the researcher, including accuracy, precision, recall, and F1 score [28].

Arabic Dialect Challenges

Studies working on ADI encounter various challenges when attempting to classify dialects. These challenges can be grouped as resource challenges and dialect differences described in the following sections.

1. Resource Challenges: Arabic is considered a low-resource language due to the lack of resources such as available datasets and tools.

- **Corpora:** The availability of benchmarks for Arabic is minimal compared to other languages such as English [24]. In English, there are various benchmark datasets to perform different NLP tasks, for instance, *SuperGLUE* and *SQuAD* [30], [31]. The former provides nine natural language understanding tasks, and the latter provides a question-answering task [30], [31]. Also, *Penn Treebank* is publicly available and provides the data with the Part Of Speech (POS) tagging on each word with a size of 4.5 million words [32].

However, Arabic dialects corpora and annotated corpora remain minimal com-

pared to MSA [25], [24]. Although a few efforts have focused on dialectal Arabic and building resources for it [33], [13], [18], [34], [4]. These datasets remain limited in size, coverage, and annotation efforts compared to MSA, and other languages [25], [24]. For example, the Arabic Treebank dataset that contains MSA consists of 243,117 words, which is around 5% of the English Penn Treebank [35]. Additionally, the parallel text for MSA with other languages is greater than dialectal Arabic where for example, the news translation is done in MSA [36], [24]. Other Dialectal-English and Dialectal-MSA are not publicly available [37], [38].

- **Linguistic Code-switching (LCD):** This phenomenon occurs when speakers alternate between languages or language varieties [39]. It is highly noticeable in online Arabic content, where most of the data is collected from [25].
 - **Mixing Arabic Varieties** Arabic speakers can switch between Arabic dialects in the same segment [39]. The presence of LCS happens between MSA and dialectal Arabic or between different dialects in a single paragraph or a sentence making dialect identification a very challenging task [25].
 - **Use of English Script** The LCD occurs in the script where the English script is used to write Arabic [25]. Although most Arabic users write in Arabic script, some users write in English script to write DA in the non-standard format of romanization that consists of numeric digits, symbols, and Latin characters that are sometimes referred to as Arabizi [25].

There are different ways the Arabic letters are transliterated; however, some Arabic letters do not have a similar phonetic approximation in the Latin alphabet [25]. Hence, these letters are expressed using numeric digits or by combining two different Latin characters making the dialect identification task in the written text very challenging due to its inconsistency and ambiguity [40], [41].

- **Tools:** The available tools for conducting different linguistic analyses remain minimal for Arabic compared to English, where English has several well-established and maintained tools for processing text such as *Stanford NLP* and *NLTK* toolkit [42]. Both are open-source tools that enable the users to perform different processing tasks on text, including POS tagging, tokenizing, and sentence splitting [42]. Additionally, the available tools for Arabic focus on a certain variety over other varieties such as *Farasa* [43], and *MADAMIRA* [44]. Although both are developed for Arabic, the former supports MSA and the latter supports a single additional variety that is Egyptian, lacking support for other Arabic varieties.

2. Dialectal Differences: The differences between Arabic varieties pose a serious challenge [24]. The differences in the vowel system among the Arabic varieties make it harder to reflect these differences in written text to identify the dialect [25]. Also, the shared vocabulary among Arabic dialects may hold different meanings. For example, the word هين can mean in Lebanese dialect *here* and another variety *ease up*. Additionally, the absence of critical marks, where some of the dialectal differences are

obvious vocally compared to written text, makes the dialect identification task more challenging [25]. Also, some Arabic letters are written in a similar manner across Arabic dialects; however, they are pronounced differently across these dialects, masking the actual pronunciation of those letters in written text [45], [46], [47], [48]. For example, the Arabic letter (ق, qaf) is pronounced differently across Arabic dialects, so depending only on the written letters can increase the error in the dialect identification task [25].

Arabic Dialect Applications

Identifying Arabic dialect can be helpful in both the linguistic and computational domains and in several applications [4]:

- Recognizing dialectal Arabic from non-dialectal Arabic can help in developing a monolingual dialectal dataset [4]. Having such data can help in several NLP systems that handle dialectal content, such as developing a dialectal language model that can be utilized in a speech recognition system [49]. Additionally, identifying Arabic dialects can help in building a parallel machine translation corpus with a dialectal part [4].
- Users are often interested in specific dialectal content, or even non-dialectal content [4]. Identifying the dialects helps provide personalized search engine results or better user-generated advertising [4]. Similarly, being able to recognize dialectal content in the user-generated content can help in identifying the user and their demographic attributes that can lead to providing better services for

these demographics [50].

- In a specific application context, such as machine translation, being able to identify the dialects can be helpful [4]. Almost all machine translation systems suffer from the Out-Of-Vocabulary (OOV) problem, which poses a serious issue [4]. In such systems, when faced with this problem, it is mitigated by discarding the words or making an extra effort to transliterate them [4]. These systems can benefit from identifying if a sentence is dialectal or not as a first step. Then it could look up the MSA equivalent which would be a much easier and more effective process [51]. Also, for problems not related to OOV, identifying the dialect can help in the polysemous word problem in dialects.

2.2 Language Modeling

The process of assigning a probability to a sequence of words in sentences in a language is known as *language modeling*. Language modeling provides a feature representation for the text by assigning a probability to the likelihood of a word following a sequence of previous words. The Language Model (LM) is considered effective if it can predict the next word in a sequence with several guesses equal to or lower than the number of guesses that the human participant delivers. This is considered an indication of human-level intelligence since predicting the next word in a sequence requires knowledge of both the syntax and semantic rules of a language in addition to world knowledge [52].

To illustrate the need for LMs, consider the case of recognition applications. It is

critical for such a product to recognize the user’s speech. If the user, for instance, said *I am happy*, the recognizer can perceive such sentence as *I am hobby*, which is not correct. Knowing that *happy* is the more probable sequence than *hobby* enables the recognizer to perform better. This applies to spelling correction and grammatical error correction as well [53]. Although currently, language models are not achieving human-level performance, it is considered a major component in different NLP applications [52].

The work on developing a representation for the language utilizes different techniques such as N-gram models [54]. In this study, we categorize the techniques as traditional and contextual LMs. Traditional models usually follow a mathematical-statistical or distributed representation approach. N-gram models follow a local representation approach, a one-to-one correspondence between concepts and representation elements. In recent years contextual representation LMs have gained wide attention. The following sections provide a description of these models.

2.2.1 Statistical Language Models

Statistical LMs are built using different techniques anchored with varying degrees in mathematical statistics [55]. These methods extract the frequency counts and the probabilities of occurrence of certain words or sentences from the training text to capture some knowledge from the text at hand. The following section covers the first proposed and the most prominent statistical LM, the N-gram method.

N-Gram

The N-gram model is the simplest LM model and is a generative-based model. It assigns probabilities to the next word in the sentence based on the n previous words.

The N-gram is a sequence of n words, where the 2-gram model, also known as *bigram* model, is a two-word sequence, for instance, *Go to* and *Do your*. The 3-gram model, also known as *trigram* model, is a three-word sequence, for instance, *Go to your* and *Do your homework* [53]. In order to compute the probability of a word w given a previous sequence of words s assuming s is “*and tried to open the bottle but*” and the aim is to know the probability of the next word *it*:

$$P (it \mid and \ tried \ to \ open \ the \ bottle \ but). \quad (2.1)$$

This can be estimated by obtaining a very large corpus and computing the number of times encountering “*and tried to open the bottle but*” and the number of times it is followed by *it* as follows.

$$P (it \mid and \ tried \ to \ open \ the \ bottle \ but) = \frac{C (and \ tried \ to \ open \ the \ bottle \ but \ it)}{C (and \ tried \ to \ open \ the \ bottle \ but)} \quad (2.2)$$

Then the probability is calculated using the *chain rule of probability* as Eq. 2.3 shows.

$$= \prod_{k=1}^n P (w_k \mid w_{1:k-1}) \quad (2.3)$$

However, the problem with the approach is that it does not hold with the growing

size of the processed text. For example, all the web text and even if it is possible, language is constantly growing and changing, and new sentences are created [53]. Hence, there is a need for an easier approach, rather than doing a complete estimate of the previous words in s given w . This is compensated by computing an approximate estimation of a given word based on the n previous words, and this is known as *bigram* and *trigram* models using the *Markov assumption* as seen in Eq. 2.4. *Markov assumption* is the probability of a word depending on the previous word.

$$P(w_{1:n}) \approx \prod_{k=1}^n P(w_k | w_{k-1}) \quad (2.4)$$

N-gram models capture syntactic information, such as the fact that *walk* is usually followed by either a noun or an adjective, or the fact that *to* is usually followed by a verb. Other information that can be captured is cultural rather than linguistic, such as people are more likely to look for Chinese food than English food [53]. Additionally, n-gram models fail to capture long-range dependencies.

2.2.2 Distribution Representation Language Models

Early linguists proposed the idea that the meaning of a word in a language comes from its distribution, the neighbor words, or the grammatical environments in the language [56] [57] [58]. Distributional representation LMs represent each attribute by a pattern of activation over multiple elements, and each computing element is involved in representing multiple entities [29]. In other words, it represents a many-to-many relationship between two types of representation, such as concepts and neurons

[59]. Where the representation of each concept is by many neurons, and each neuron participates in the representation of many concepts [59].

In NLP, this concept is represented by using vectors where a vector represents each word, and this is called *embeddings* [53]. These embeddings are a powerful component in NLP applications such as sentiment analysis. The following section describes one of the most commonly used models, *Word2Vec*, which constructs short and dense vectors with some helpful semantic proprieties [53].

Word2Vec

Word2Vec is a dense and short vector with d dimension that can range from 500-1,000 vocabulary size $|V|$ and the vector values are real-valued numbers that can be either positive or negative numbers [60], [53]. Word2Vec mitigated the problem of long, sparse, and mostly-zero valued vectors, and it performed better than sparse vectors. In addition, it can perform better in capturing synonyms. The reason is not completely known, but the intuition is that the smaller vector size (300 dimensions) enables the classifiers to learn fewer weights than sparse vectors that can have vectors with far more weights (50,000 dimensions). In addition, the smaller parameter space might help generalize the models to avoid overfitting [53].

Word2Vec embedding is a static embedding, which means that each word in the vocabulary is learned in one fixed embedding in contrast to contextual embeddings that dynamically learn embeddings discussed in Section 2.3 [53]. The computed vectors showed that this can capture many linguistic regularities; for example, vector operations $vector('Paris') - vector('France') + vector('Italy')$ result in a vector that

is similar to $vector('Rome')$, and $vector('king') - vector('man') + vector('woman')$ is similar to $vector('queen')$ [61] [60] [62]. Word2Vec can be computed using either Skip-gram or Continuous Bag of Words (CBOW) algorithms [63].

In CBOW, the order of words in this architecture does not matter, which explains calling it bag-of words [64]. CBOW predicts the target word (middle word) based on the surrounding context words [64]. The context is the words that come before and after the current (middle) word, as Figure 2.6 illustrates. This architecture is faster than the Skip-gram discussed below, and it can be better suited for small datasets [64].

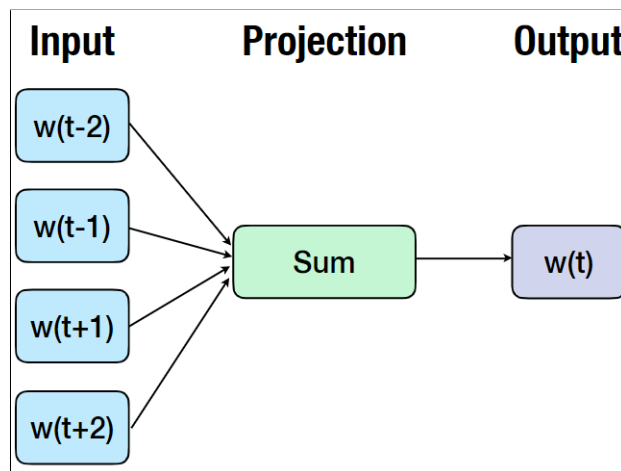


Figure 2.6: Word2Vec: CBOW architecture.

One method for computing the embeddings is using Skip-gram with Negative Sampling (SGNS) [53]. Skip-gram predicts the neighboring words around the current word with a specified context window before and after it in the same sentence, as shown in Figure 2.7 [64]. The intuition is as follows [53]:

1. Consider both the target word and the neighboring words representing the context as a positive example.

2. The negative samples are a random sampling from a lexicon built from the corpora.
3. Classify the positive and negative samples by training a logistic regression.
4. The learned weights represent the embeddings.

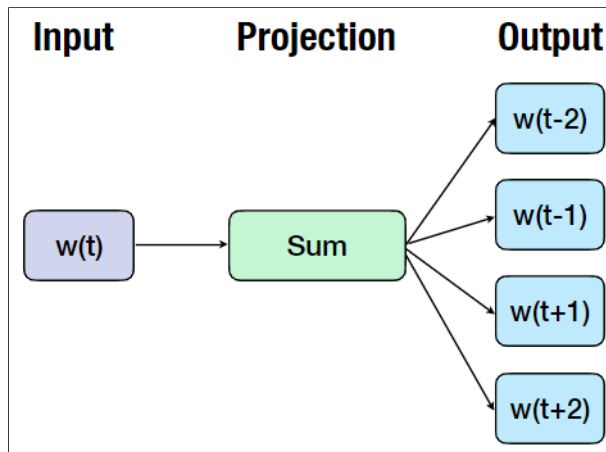


Figure 2.7: Word2Vec: Skip-gram architecture.

2.3 Contextual Representation Language Models

The word meaning in a language often relates to its neighboring words in the sentence or what is formally known as the word context. The word *mouse* in the English language can have two meanings, the animal and the computer mouse. Providing the contextualized representation for such a word in a language can lead to better performance in different tasks [29]. ELMO is the first model that provides the contextualized representation of words [65]. However, it lacks the bidirectional understanding of a language. Consequently, Bidirectional Encoder Representations from Transformers (BERT) is introduced to provide a bidirectional contextualized representation of

words in the language [66]. This section covers BERT components since an optimized version of BERT is utilized in this work.

2.3.1 Encoder-Decoder Networks

The idea of the encoder-decoder networks or sequence-to-sequence models is to generate an output sequence that represents the context properly. The encoder network takes a sequence as an input and develops a contextual representation of that sequence [53]. Then the decoder network takes that representation as an input and generates a task-specific output [53]. This architecture started from machine translation; the goal of this task is to take an input sequence from a *source* language and output a *target* sequence from another language. The training text in this task is known as *bitexts* or *parallel texts*, consisting of pairs of sentences from different languages that are a translation of each other [53].

Before starting the machine translation task, an end-of-sequence marker $</s>$ is added at the end of each source-target sequence pair, and these pairs represent the training data for the language model. The Recurrent Neural Network (RNN)-based language model trains the network to predict the next word in a sequence; this type of language modeling is referred to as *autoregressive modeling*. In machine translation, as Figure 2.8 shows, the encoder network performs forward inference generating hidden states until the end of the source sentence sequence is reached. Then the autoregressive generation process starts in the decoder by taking the end-of-sequence marker, and a word in the context of the hidden layer from the end of the source input [53]. The following words are conditioned on the embeddings of the last generated word and the

previously hidden state [53].

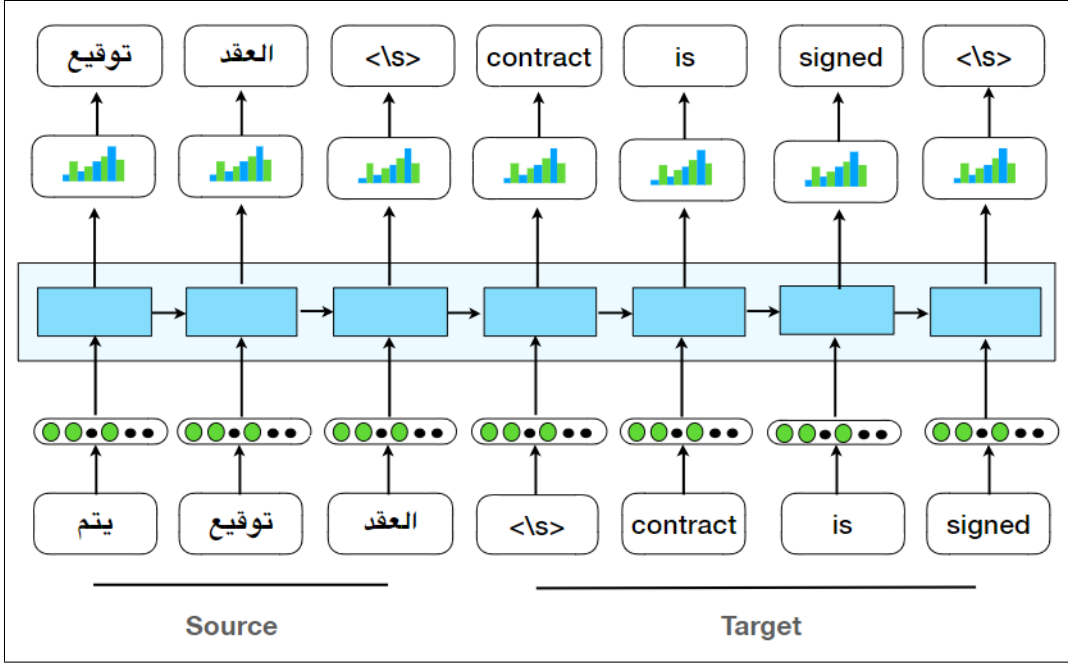


Figure 2.8: The training process for a neural language modeling approach for machine translation. The pair bitexts are concatenated and used to train a language model.

Abstracting from the machine translation task, Figure 2.9 illustrates the encoder-decoder network. The left part of the figure shows the *encoder* that generates a contextual representation of the input sequence [53]. This representation takes from the final hidden state of the encoder h_n and feeds it to the decoder. The *decoder*, the right part in the figure, then takes the final state and autoregressively generates a sequence of outputs [53].

This architecture suffers from major design issues. First, both the encoder-decoder are assumed to have similar internal components. Also, the final hidden state of the encoder is the only context available to the decoder as the initial hidden state [53]. The basic architecture of the encoder-decoder, regardless of the application domain and the internal components, consists of three main components [53]:

- **Encoder:** Takes as an input a sequence x_1^n and generates the contextual representation h_1^n .
- **Context Vector:** Denoted by c is a function of the h_1^n that transfers the meaning of the input to the decoder.
- **Decoder:** Takes c as an input and generates inconsistent length sequence of hidden states h_1^m . From these states, a corresponding output state sequence y_1^m is obtained.

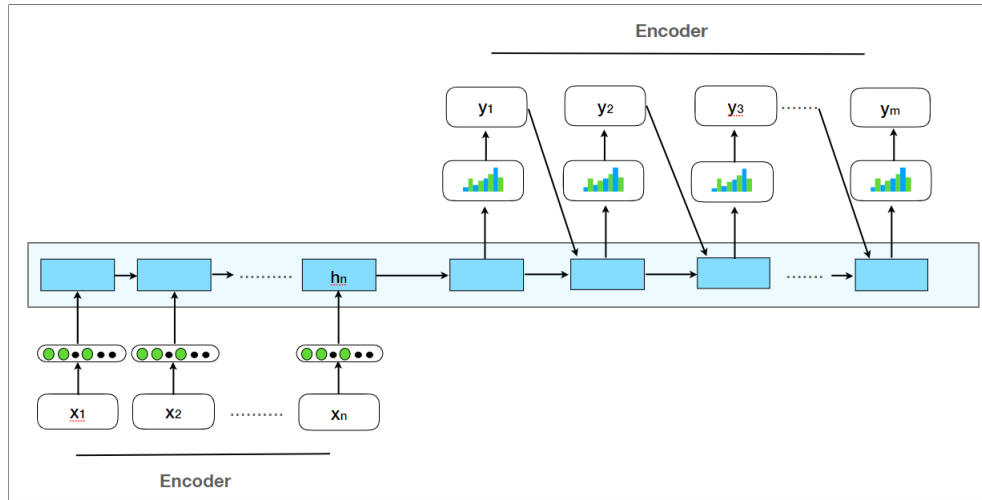


Figure 2.9: Simple RNN-based encoder-decoder architecture. The RNN encoder provides the decoder with the final hidden state as the context.

2.3.2 Attention

The downside of the traditional encoder-decoder architecture is in the context available to the decoder since it takes only the last hidden state of the encoder. In order to mitigate this problem, the full encoder context c needs to be considered by the decoder

during the decoding process in a dynamic manner and represented to it in a fixed-size vector [53]. The *attention mechanism* resolves this problem.

Attention replaces the static context vector with a dynamic context vector taken from the encoder’s hidden states at every point during the decoding process and provides it to the decoder dynamically. This context vector c_i is generated at each step i and considers all the encoder hidden states at the generation process. This context is then available during the decoding process by conditioning the current decoder state computation on it[53]. In addition to the previous hidden state h_{i-1}^d and the previous output generated by the decoder $\hat{y}_{i-1}$ [53].

$$h_i^d = g(\hat{y}_{i-1}, h_{i-1}^d, c_i)$$

To compute the context c_i , we need to calculate a score between each encoder hidden state and the decoder hidden state to capture the most relevant encoder hidden state to the captured decoder hidden state h_{i-1}^d . This is done during each state i of the decoding process the score $score(h_{i-1}^d, h_{i-1}^e)$ for each encoder state j [53].

$$score(h_{i-1}^d, h_{i-1}^e) = h_{i-1}^d \cdot h_{i-1}^e$$

The scalar result score represents the degree of similarity between the two vectors, and this provides us with the relevance of each encoder state to the decoder state at each time step. However, the problem here is that a scalar value represents the similarity value. Although it is very effective in representing the similarity, it is not easy to adopt in the training process. Hence, a more robust similarity score can be obtained by having a set of weights W_s . Having the weights W_s enables the network to learn the important encoder states to the decoder in the current application [53].

$$score(h_{i-1}^d, h_{i-1}^e) = h_{i-1}^d W_s h_{i-1}^e$$

To produce a vector of weights α_{ij} , a softmax is used to normalize the weight scores. This enables us to identify the relevance of each encoder hidden state j to the current decoder state, i .

$$\begin{aligned}\alpha_{ij} &= \text{softmax} \left(\text{score} \left(h_{i-1}^d, h_j^e \right) \forall j \in e \right) \\ &= \frac{\exp(\text{score}(h_{i-1}^d, h_j^e))}{\sum_k \exp(\text{score}(h_{i-1}^d, h_k^e))},\end{aligned}$$

Finally, to compute a fixed-length context vector that represents the current decoder state, we take the weighted average over all the encoder hidden states given the distribution α [53].

$$c_i = \sum_j \alpha_{ij} h_j^e$$

2.3.3 Self-Attention and Transformer Networks

Sequence modeling is one of the well-known techniques in neural contextual-based language modeling that captures the word local context representation; the RNNs and RNN-based models are primarily used in sequence modeling [67]. These models cannot perform parallelization due to the recurrent nature [68]. Also, they require long computation time since it generates a sequence of hidden states h_t as a function of the previous hidden state h_{t-1} and the input from position t [68].

The attention mechanism became an essential technique in sequence modeling since it allows the modeling of long sequences without being concerned about its distance from the input or output sequences [68]. The attention mechanism was used with recurrent networks until Transformers were introduced, which are the first fully attention-based encoder-decoder architecture [68]. Transformers removed the recurrence entirely and drew global dependencies between input and output sequences

allowing for parallelization and reducing the computation time without lowering the result [68]. On the contrary, it achieves state-of-the-art performance on the translation task [68].

Transformers are fully built on *Self-Attention* that eliminates recurrence entirely [68]. Transformers are composed of encoders and decoders stacked on top of each other. However, to explain Transformers, for explanation the translation task is used as an example with a simple encoder and decoder as Figure 2.10 shows.

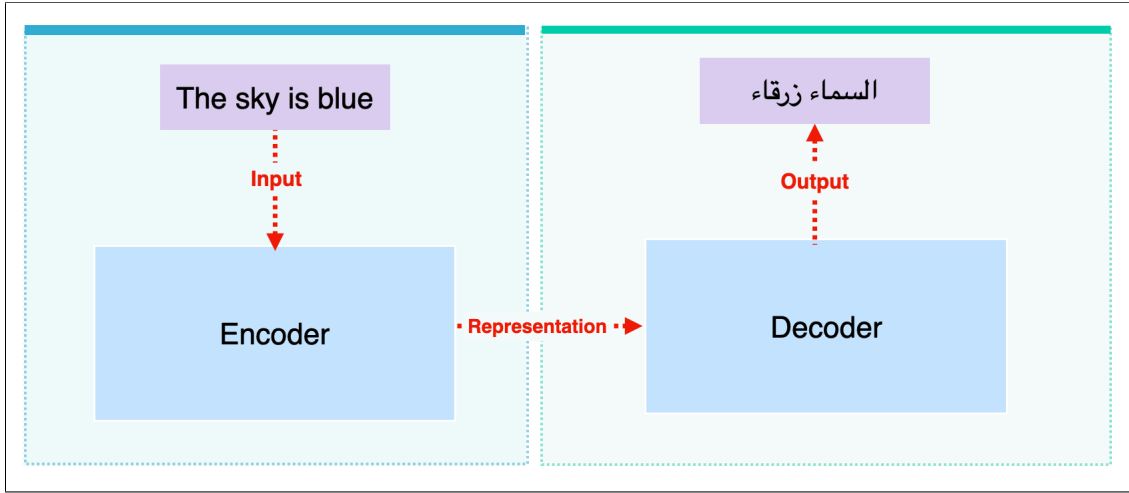


Figure 2.10: The encoder-decoder in a Transformer.

In the case of translation from English to Arabic, for example, the input to the encoder is a sentence (source sequence). The encoder learns the representation of that sentence and outputs the representation as an input to the decoder. The decoder then generates the output of the Arabic sentence (target sequence). This represents an abstract view of the inner workings of the Transformer illustrated below.

Figure 2.11 depicts the proposed Transformer model architecture. Transformer architecture encapsulates various components, including positional embeddings, encoders, and decoders, each serving and operating differently. The encoder and decoder

share several main inner components; however, we can divide the component based on functionality.

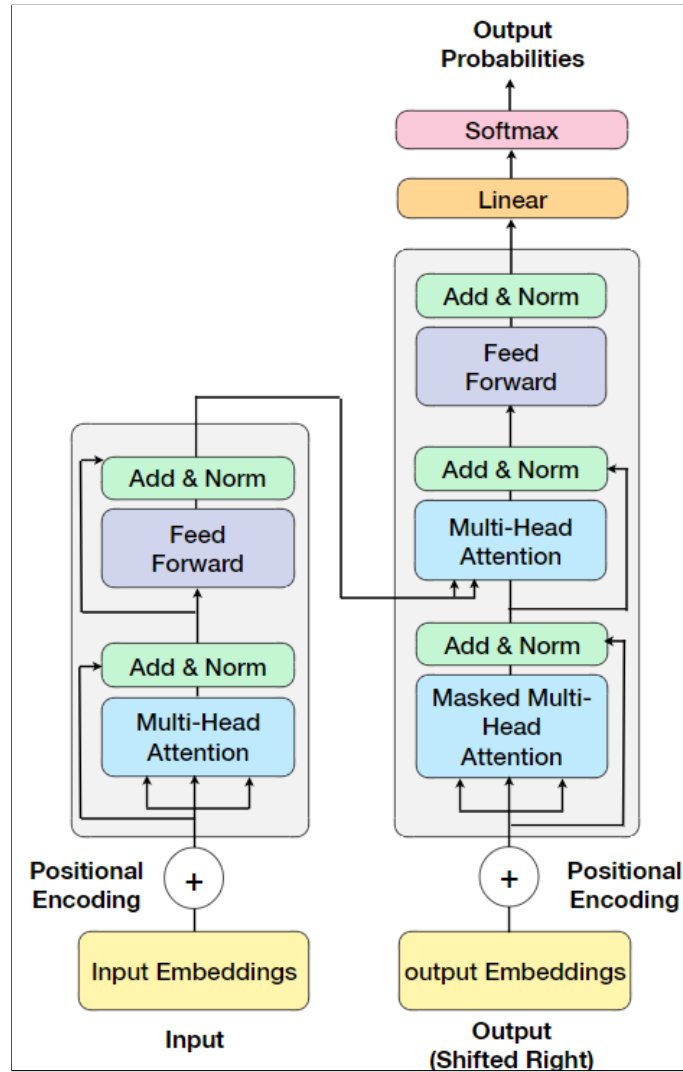


Figure 2.11: Transformer architecture.

- Handling both the input data to the encoder and output data from the decoder allows the model to learn relative positioning.
 - **Embeddings:** it contains both the *embedding layer* and the *position encoding layer*.

- Generating a contextual representation from the input and feeding it to the decoder. This is done by using a stack of N encoders, usually *six*; it encapsulates different components that provide different functions.
 - **Encoder:** it contains *Multi-Head Attention* sublayer, *Feed-forward* sublayer, normalization layer and residual connection.
- Generating an output sequence that represents the context fed by the input to the decoder. This is done by using a stack of N identical decoders, usually *six*, that encapsulate different components that provide different functions.
 - **Decoder:** it is similar to the encoder components and includes: *Multi-Head Attention* sublayer, *Feed-forward* sublayer, normalization layer, residual connection and adds a third sublayer of *Masked-Multi-headed Attention*.
- Training optimization to stabilize performance and reduce training time.
 - **Residual connection.**
 - **Normalization Layer.**
- Generating output probabilities providing the final user output for a given task.
 - **Linear layer.**
 - **Softmax layer.**

The following section describes each component to understand the Transformer model's inter-workings.

Embeddings

In order to provide a meaningful representation of words to the Transformer model, two things need to be captured by the model: the word meaning and position among the input sequence. The embeddings that are also known as vector semantics provide a vector representation of the word from its context [53]. The positional encoding is significant to learn since the attention mechanism lacks recurrence and convolution; the model can utilize the order of the sequence [68].

1. **Embedding Layer** The embedding layer is present at both the encoder and the decoder as it is used for neural networks on text data to encode unique numeric values [69]. First, the input sequence is tokenized; each sentence is split into words. The tokenizers follow different methods for tokenization; however, the outcome is usually similar [70]. Then each word is represented with a unique random number that is then represented by vector weights that are randomly initialized. Finally, the embeddings are learned from other data during the training phase [69].
2. **Positional Embedding** The significance of Transformers lies in their ability to train in parallel to alleviate the challenges of recurrence. However, it introduces another challenge where the Transformers cannot figure out the position of a word in a sequence, and introducing a positional vector is computationally expensive [70]. This is resolved by adding a *positional encoding* function that calculates a value according to the position of the word and adds it to the input embedding [68]. The positional encoding is calculated using a *sine* and *cosine*

function that generates different frequencies for the *positional encoding* as shown below [68]. It has the same dimension of the embedding vector d_{model} to be able to add it to it [68].

$$PE_{(pos,2i)} = \sin \left(pos / 10000^{2i/d_{model}} \right)$$

$$PE_{(pos,2i+1)} = \cos \left(pos / 10000^{2i/d_{model}} \right)$$

The pos represents the position, and i is the dimension. The dimension of the *positional encoding* makes the model learn easily to attend to relative positions for any k in PE_{pos+k} and can be represented as a linear function of PE_{pos} .

Encoder and Decoder

The encoder-decoder in the Transformers consists of four components: multi-headed self-attention, feed-forward network, residual connection, and normalization layer. The following section describes these components.

1. **Multi-headed self-attention** is based on *self-attention* mechanism [71]. Before explaining *Multi-head attention* we will explain *self-attention*. Assume having the following sentence ***The animal didn't cross the street because it was too tired.*** In this sentence the ***it*** can refer to either the ***animal*** or the ***street***. As readers, it is known that ***it*** refers to the ***animal***, not the ***street***. However, the model does not refer to that and here *self-attention* can help. A model with *self-attention* computes the representation of each given token in the input sequence. During the computation of the representation for each token, it relates the token to all of the other tokens in the input sequence as shown in Figure 2.12. By relating the token ***it*** to all of the other tokens, it understands

that *it* is related to the *animal* token, not the *street* token. Hence, the line that relates *it* to *animal* in the figure is the thickest line.

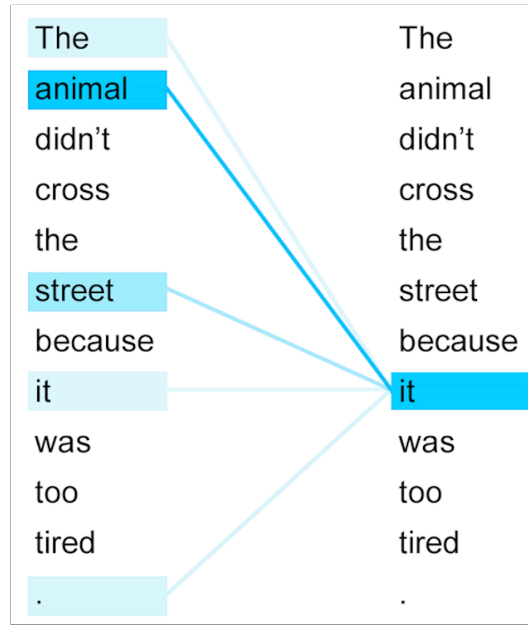


Figure 2.12: Self-Attention Example [2].

The implementation details show that the input sequence is taken as an embedding where each token is represented as a vector. Let x_1 be the representation vector of $token_1$ and x_n represents $token_n$. Then the input matrix (X) of size $n \times 512$ since n represents the input sequence number and the embedding dimension is 512. After taking the input matrix X , attention requires creating three new matrices: query matrix Q , the key matrix K , and value matrix V . In order to create these matrices, another three new weight matrices are created: W^Q , W^K , and W^V . The weight matrices are randomly initialized and multiplied by the input matrix X to produce query, key, and value matrices.

These three matrices are used to calculate *self-attention*, summarized in four main steps [71]:

- (a) **Step 1:** Compute the dot product between the query matrix Q and the transposed key matrix K^T , to obtain the similarity matrix QK^T .
- (b) **Step 2:** Divide QK^T matrix by the square root of the dimension of the key vector $\sqrt{d_K}$, to obtain stable gradients.
- (c) **Step 3:** Apply the softmax function on the resulting matrix from the previous step to produce the score matrix $\text{softmax}(QK^T/d_K)$. This is done to obtain a normalized version of the matrix where the values are between 1 and 0.
- (d) **Step 4:** Compute the attention matrix Z by multiplying the score matrix $\text{softmax}(QK^T/d_K)$ by the value matrix V .

Now we address *Multi-head attention*, there is a need for it since using *self-attention* can lead to an ambiguous understanding of the word. To avoid this problem, an attention matrix is created by calculating different attention matrices Z . For example, having six attention matrices Z_1 to Z_6 . These matrices are concatenated and multiplied by a new weight matrix W_0 , leading to a final attention matrix as shown below:

$$\mathbf{Multihead\ attention} = \mathbf{Concatenate} (Z_1, Z_2, \dots Z_i, \dots Z_8) W_0$$

2. **Feed-forward layer** is a sublayer in the encoder as shown in Figure 2.11. The feedforward network consists of two fully connected layers where all the neurons are connected with the ReLU activation function, and the output of each layer is passed to the following layer [53], [71].

3. **Residual Connection**, also known as *Residual Learning*, is first introduced in computer vision. It makes the process of training deep networks faster by reformulating the layers and learning from them rather than using another random function [72].
4. **Normalization Layer** also allows for faster training by not allowing the values to change in a significant manner at each layer [71].

Output Probabilities

The learned representation is then fed through linear and softmax layers to obtain the output [71].

1. **Linear layer:** this layer generates the logits that have the same size as the vocabulary [71].
2. **Softmax layer:** the generated logits are then converted to probabilities in the softmax layer using the softmax function [71].

2.3.4 Bidirectional Encoder Representation from Transformers

Bidirectional Encoder Representation from Transformers (BERT) is a new language representation model designed to represent text from both left-to-right and right-to-left directions [66]. BERT reduces the need for heavy architecture engineering required when working on a specific task by using a pre-train deep bidirectional representation on unlabelled text [66]. This results in a developed model that can be utilized by

transfer learning on different downstream tasks creating a state-of-the-art model by only adding one additional output layer without the need to make a substantial change in the model architecture [66].

Model Architecture

The BERT model is based on Transformers. As discussed in Section 2.3.3 transformers are composed of encoders and decoders. However, BERT is built based on only the encoder part of the Transformer [71]. As described earlier, the encoder takes the sentence as input, and it returns the embedding representation of each word as output, and this is the primary function of BERT [71]. Previous to BERT, the representation of a sentence was unidirectional, where a component represents each representation, either left-to-right or right-to-left representation [66]. In order to capture the bidirectional context, the models concatenate both representations; however, this is not the case with BERT [66]. BERT provides the bidirectional representation of the sentence without the overhead of additional components [66]. In order to understand BERT let us take an example sentence '*NLP is great*' and feed it as an input to the encoder of the Transformer as Figure 2.13 illustrates. Then the stack of encoders yields the contextual representation of each word as an output embedding. The embeddings are developed by feeding the sentence to the encoder, and then *multi-head attention* relates each word to the other words in the sentence. This step enables learning the contextual relationship of each word with its context by capturing the contextual semantic meaning of the word and representing it as an output embedding [71].

The size of the output representation for each word is dependent on the size of the encoder layer; the proposed size is 768 [66], [71]. Figure 2.13 shows N number of

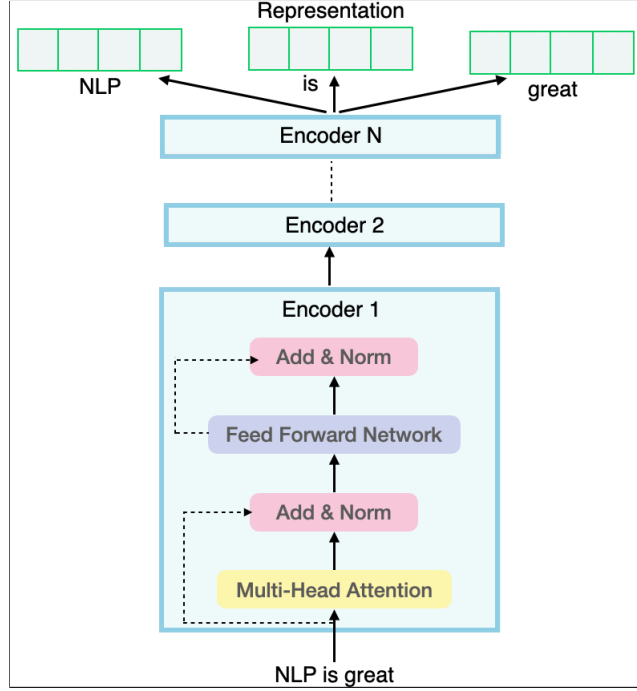


Figure 2.13: BERT provides the embedding representation of an example sentence.

encoders; since the number of encoders is not fixed it depends on the utilized BERT configuration [71]. There are two standard configurations for BERT either *BERT-base* or *BERT-large* [66]. BERT-base consists of 12 encoder layers, 12 attention heads and 768 hidden units [71]. BERT-large consists of 24 encoder layers, 16 attention heads and 1024 hidden units [71]

BERT operates in two different settings, the learning phase, and the transfer learning phase, as Figure 2.14 illustrates [66]. During the learning phase, the language model is developed, that is the contextual representation that is utilized in the transfer learning phase on the specified downstream task [66]. The following section describes these two phases.

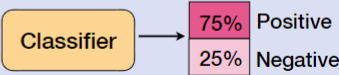
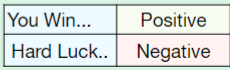
Phase	Self-Supervised Learning		Supervised Learning	
Objective	Predict	MLM or NSP	Classify	
Model	Training		Pre-trained	
Dataset	Unlabeled Text   		Labeled Text	

Figure 2.14: BERT learning settings (1) Learning: self-supervised trained on large textual datasets. (2) Transfer learning: supervised learning on a specific downstream task with an annotated dataset.

Learning In BERT

The learning of BERT is the process where the model is learning from a large dataset with a specific objective and saving the developed model with the final reached weights [71]. This then can be used when a new task needs investigation; rather than learning from scratch, the new task utilizes the learned weights [66]. In BERT, there are two learning strategies: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP), followed by the transfer learning process [71]. The process for representing the input before processing it is provided below, followed by the learning strategies.

1. **Input Data Representation:** BERT takes the input as an embedding representation that is built by three different embedding layers as indicated below. Before feeding the input, two tokens are added, a [CLS] token indicating the start of a sentence and a [SEP] token indicating the end of the sentence [71].

- **WordPiece Tokenizer:** before feeding the text to the different input layers, it is first tokenized. Tokenization is the process of segmenting a body of text into words [53]. BERT uses the *WordPiece tokenizer* to tokenize text

since it handles the OOV problem [71]. This algorithm requires defining the vocabulary size, and then it starts breaking text into tokens and checks if the token already exists or not [71]. In case the token does not exist, then the algorithm breaks the token to sub-tokens and checks again until it reaches individual characters [71]. Then these tokens are fed to the three following layers to provide the input embeddings.

- **Token Embedding:** in this layer, the input words that are also known as tokens are converted into embedding as shown in Figure 2.15 where for example, $E_{[CLS]}$ represents the embedding for [CLS] token.
- **Segment Embedding:** in this layer, the segment indicates the sentence to distinguish if a token belongs to sentence A or B as shown in Figure 2.15 where the token is mapped to E_A if the token belongs to sentence A.
- **Positional Embedding:** Transformers process tokens in parallel, contrary to the recurrence-based methods that process tokens sequentially. This requires a mechanism to encode the positional encoding; this is done by adding the positional encoding layer to indicate the position of the token in the sentence, as illustrated in Figure 2.15.

After converting the input to token embedding, segment embedding, and positional embedding, they are all added together and fed to the BERT model as illustrated in Figure 2.15.

2. **Learning Strategies:** BERT is developed to learn on two tasks: *MLM* and *NSP*. These learning strategies are explained below.

Positional Embeddings	E ₀	E ₁	E ₂	E ₃	E ₄	E ₅	E ₆	E ₇	E ₈
Segment Embeddings	E _A	E _A	E _A	E _A	E _A	E _B	E _B	E _B	E _B
Token Embeddings	E _[CLS]	E _{Token1}	E _{Token2}	E _{Token_n}	E _[SEP]	E _{Token1}	E _{Token2}	E _{Token_m}	E _[SEP]
Input Text	[CLS]	Token ₁	Token ₂	Token _n	[SEP]	Token ₁	Token ₂	Token _m	[SEP]

Figure 2.15: BERT input embedding representation.

- **Masked Language Modeling:** Prior to explaining MLM the language modeling tasks need to be explained. Language modeling can be *Auto-regressive* or *Auto-encoding* [71]. Auto-regressive is based on the prediction of the next word either in a *forward* or a *backward* manner; forward refers to the prediction of the next word from left-to-right and backward from right-to-left [71]. Auto-encoding language modeling reads the sentence from both directions in order to make a prediction, making it bidirectional in nature [71].

BERT uses auto-encoding language modeling; it reads the sentence from both directions and makes the prediction [71]. In MLM used in BERT, 15% of the input sentence is randomly masked with a [MASK] token to train the networks to predict the masked tokens [71]. However, masking the full 15% of the input leads to a variation between the learning and transfer learning phases [71]. Hence, this approach is optimized by following the 80-10-10 approach where 80% of the 15% is randomly masked, 10% is replaced random token, and the final 10% is not changed [71]. After tokenizing and masking the text, it is fed through the input embedding layers.

After feeding the input to the encoders, it yields the embeddings as illustrated earlier in Figure 2.13, and to predict the masked tokens, the contextual embeddings are utilized [71]. The produced $E_{[MASK]}$ by BERT is then fed to a feedforward network with a softmax activation function [71]. The network then produces a probability of all the tokens in the vocabulary to replace the masked token [71]. In the initial iterations of BERT, the probabilities of the words to replace the masked token were not accurate since the weights are not optimal yet [71]. However, with more iterations and with the back-propagation, the weights will reach a near-optimal state producing more accurate predictions [71].

- **Next Sentence Prediction** is another learning strategy in the BERT model; NSP is a binary classification task [71] where two sentences (pair) are fed to the model, and BERT has to predict if the following sentence is the subsequent sentence for the previously fed sentence or not [71]. In this task, the model can understand the relationship between the pair of sentences, and this is useful for some of the downstream tasks such as question answering and text generation [71].

Transfer Learning In BERT

After learning in the BERT model, the weights are saved and utilized in different downstream tasks without the need to build a new model architecture [71]. These tasks include text classification, natural language inference, and question-answering as Figure 2.16 illustrates [71]. During the transfer learning phase, the previously

developed embeddings are updated according to the task at hand rather than learning from scratch [71]. Transfer learning is illustrated below in the text classification task to see how to perform the task using the developed BERT.

The binary text classification task aims to classify if a text is positive or negative. Now, assume there is an available annotated dataset for the text classification task where each sentence is labeled as either positive or negative. As mentioned earlier, the embedding vector represents each word, and the [CLS] token is added at the beginning of each sentence. The $E_{[CLS]}$ holds an aggregate representation of the sentence; hence it is used for the transfer learning task rather than the embedding of each token [71]. Then the $E_{[CLS]}$ is fed to a feedforward network with a softmax function (classifier) to train the classifier to perform the classification task [71]. The weights are updated in both the developed BERT model and the classifier [71].

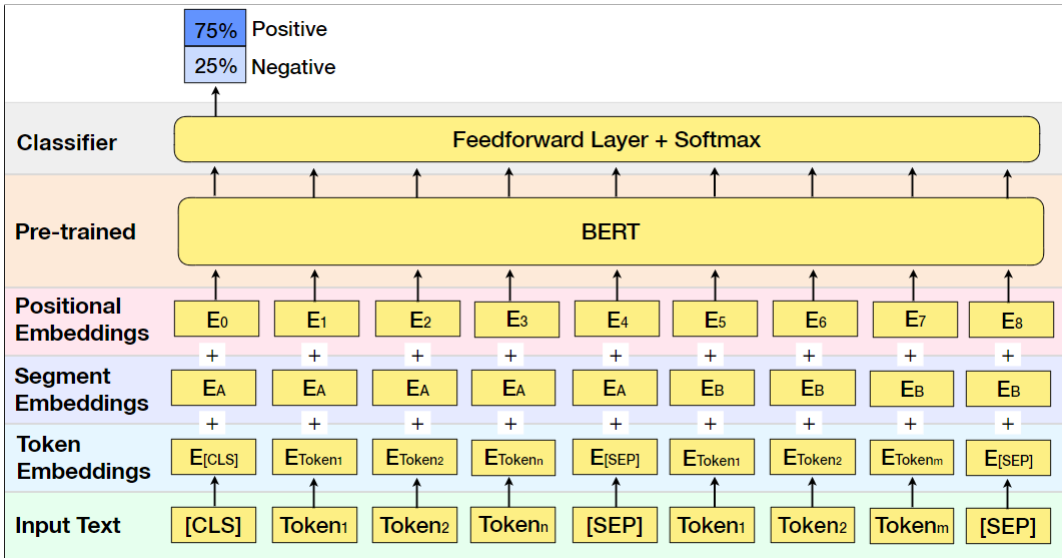


Figure 2.16: Illustrations of transfer learning in BERT on the text classification task.

CHAPTER 3

LITERATURE REVIEW

This chapter covers the surveyed literature for three main topics. The first section covers the work on Arabic dialect resources to identify the available resources. The second section covers the work on the ADI task, and the third highlights the work developing Arabic neural language modeling for non-contextual and contextual modeling.

3.1 Arabic Dialect Resources

Arabic dialect resources are not abundant as one might think. The available resources for Arabic usually focus on MSA due to its availability in formal prints such as books and newspapers, unlike dialectal content that is primarily available in online-based sources. This work concentrates on dialectal Arabic after surveying the literature summarized in Table 3.1 since there is a consensus among studies on the scarcity and the need for vernacular Arabic resources. This led several studies to propose different methods to collect and build dialectal datasets to investigate tasks such as sentiment

analysis, dialect identification, and morphological analysis [73], [16], [74], [33], [75], [8], [76]. The proposed corpora vary in terms of size, availability, source, annotation, task, and dialect, as shown in Table 3.1. The following sections describe the data sources, the Arabic corpora, and the Arabic dialects in the corpora.

3.1.1 Arabic Corpora Data Sources

Arabic dialect research works makes use of user-generated content as a main source to represent different Arabic dialects [33], [13], [77], [73], [78], [79], [74], [76]. Most of the work uses Twitter as the primary resource for building dialectal corpora. This can be attributed to the API that Twitter provides; it allows collecting data easily [80]. Also, it allows specifying different parameters when collecting data, including location, language, and keywords to collect data about, allowing researchers to have more control over the collected data [80]. Other online sources include user reviews or comments on products or services. Studies utilizing this resource mainly investigate sentiment analysis or build a corpus for a specific dialect [16], [76], [81], [82], [12].

The content from user-generated websites is from users without any formal editing, making it a rich source for dialectal Arabic. However, that does not eliminate the presence of MSA inside these datasets [15], [83], [84], [82], [78]. Also, the corpora size varies due to the data source or other limitations enforced by these data sources such as Twitter, which limits the rate of retrieved data [85]. Another limitation can be the period of collecting data; some studies collect data over a more extended period than other studies, allowing for larger corpora to be built [33], [83], [78].

3.1.2 Arabic Corpora

There is a general consequence on the scarcity of dialectal resources for Arabic, which motivated many works to build corpora either for a specific Arabic dialect or to investigate specific tasks in Arabic [15], [33], [13], [77], [16], [83]. Despite that, unfortunately, most corpora are not publicly available, limiting the works' contribution to illustrating the methodology of building the corpus for a specific dialect or task [73], [76], [14], [75], [81], [82], [12]. Other works make the built corpora publicly available, motivating more work utilizing the shared corpora [77], [13], [33], [16], [15].

Annotation of corpora differs greatly depending on factors such as the investigated task, the dialect granularity level, the annotation method, and the number of labels. The task affects the annotation process where dialect identification or sentiment analysis can be on the sentence level, and the annotation for morphological purposes is on the word level [8], [76]. Also, the dialect annotation task can be done depending on the location provided by the user or available in the metadata of the tweet and considering it as a label for that text automatically [15], [16], [12]. Human annotators can be found in various ways such as recruiting graduate students or using online services such as Amazon Turk [14].

Labels also vary between different datasets which require more annotation efforts to adopt these datasets. For example, a corpus containing region-level dialect needs another annotation effort in a study to investigate the city or the country granularity level. Finally, the size of the annotated corpora remains much smaller compared to the available unlabelled data [73], [14], [82], [79].

There is no general consensus on grouping Arabic dialects in studies. However, there are four observed trends. In the first trend, studies mention that the work is a mix of MSA and dialectal Arabic without specifying the variety of Arabic [15], [16], [82], [12], [74], [83]. The majority of this work investigates the sentiment analysis task. They collect the data from different online sources that do not provide the location information about the collected posts, making it difficult to identify the dialect granularity level [73], [16], [14], [74], [79].

The other trends differ in granularity level when investigating Arabic dialect since the Arabic language is a family of varieties. In the second trend, studies work on the coarse grind of Arabic by grouping the dialects based on the geographical region where each region consists of different countries [14], [13], [18]. However, the countries' dialects differ from one to another on all levels of linguistic representation, from phonology and morphology to the lexicon and syntax [17].

That led studies towards two alternatives when investigating Arabic dialects on a finer granularity level, the country and city level, as observed in the third and fourth trends. The third trend works on a finer-grained dialect that is the city level that studies the dialect identification task [13], [18], [8]. The final trend works on the country-level dialect to investigate different tasks, including sentiment analysis, morphological analysis, and dialect identification [8], [79], [74], [81].

Corpus	Source	Task		Available	Size		Annotation		Arabic Variety	
		DI	Other		Unlabeled	Labeled	Manual	Automatic	MSA	DA
ArCOV-19 [77]	Twitter	-	X	Link	2.7M Tweet	60,873 Tweet Geo-located 2,078 Tweet Geo-tagged	-	X	NA	NA
SS2030 [73]	Twitter	-	X	NA	-	4,200 Tweet	X	-	-	X
NADI [33]	Twitter	X	-	Link	10M Tweet	21,000 Tweet	X	-	-	X
MADAR [18]	Twitter	X	-	Link	-	2,980 Twitter user profiles	X	-	X	X
SANAD [15]	News Articles	X	-	Link	-	200K	-	X	X	-
HARD [16]	Hotel reviews	-	X	Link	-	93,700 Reviews	-	X	X	X
SAUR [76]	Twitter YouTube Whats-App Blogs Forums Instagram	-	X	NA	-	104K Word	-	X	-	X
ArSenTD-Lev [14]	Twitter	-	X	NA	-	4,000 Tweet	X	-	-	X
Multi-Dialectal Tweets [86]	Twitter	-	X	NA	250M Tweet	NA	X	-	-	X
CALYOU [81]	YouTube	-	X	NA	853K Comment	-	-	-	X	X
EveTAR [83]	Twitter	-	X	Link	355M Tweet	22K	X	-	X	X
Tashkeela [84]	Al-Maktaba Al-Shamela website	-	X	Link	75M Word	-	-	-	X	-
Arasenti-Tweet [74]	Twitter	-	X	NA	-	17,573 Tweet	X	-	X	X
Facebook Sentiment [82]	Facebook	-	X	NA	-	2,000 Post	X	-	X	X
Saudi Corpus [79]	Twitter	-	X	NA	-	4,700 Tweet	X	-	-	X
BRAD [12]	Good Reads	-	X	NA	510K Review	156K	-	X	X	X
Moroccan Sanaani Corpus [8]	Online Printed	-	X	NA	-	64K Words Moroccan 32.5K Words Yemen	X	-	-	X
Arasenti [78]	Twitter	-	X	NA	2.2M Tweet	10K Tweet	X	-	X	X

Table 3.1: Summary on the recent work on Arabic Corpora.

3.2 Arabic Dialect Identification

Identifying dialects is a more challenging task compared to the language identification task [87]. Especially for Arabic, since all Arabic dialects share identical script and part of their vocabulary is shared among the dialects [88]. This increases the difficulty of distinguishing between these dialects [88]. Consequently, development emerged in recent years to automate the process for identifying dialects [88].

3.2.1 Dialectal Granularity Level

The linguistic variation among Arabic dialects makes the granularity level an important dimension when investigating them. Arabic dialects are identified based on three levels: region, country, and city level. The research on ADI is mixed on the dialect granularity level. In early work on dialect identification, the work focused on identify-

ing the dialect on the regional level since the data source is from the user’s comments online, making it a challenging task to identify a finer grind dialect level [89], [90]. However, recent work is investigating a finer dialect level, namely country and city, utilizing Twitter as a source for collecting data where the location tag can be defined when collecting data [91], [92], [93], [94], [95], [96].

Although many studies have investigated the city-level dialect identification, the performance on this task yielded unsatisfactory results calling into question the possibility of identifying the dialect on the city level with the available data [97], [98], [99], [100], [91]. This led other studies to investigate the country level for the dialect identification task, and yielded comparable results [93], [94], [95], [96]. This might be a result of the fact that the data is collected from Twitter in all previous cases based on the geolocation provided by the user which does not necessarily indicate that the user actually speaks the city- or country-level dialect.

3.2.2 Features

Working with a written text requires transforming it into low dimensional semantic space; usually embeddings are used to represent a word in a machine-readable format [29]. A prominent approach in the literature is the n-gram based LM to represent features. Although n-gram models are heavily used across the literature on ADI tasks, no consensus appears on the value of the n-gram. The level of representation for the word and the character is based on the empirical investigation for each study. Although several reported using the unigram model on the word level yields good results in Arabic dialect, however, they have adopted different character-level n-gram

representations due to the Out Of Vocabulary (OOV) problem [91], [97], [92], [98].

The OOV poses a serious challenge when working on Arabic dialects since Arabic is a morphologically rich language, and the dialects differ from each other as depicted earlier [6], leading studies to experiment with more n-gram models on the word level ranging from unigram models to 16-gram models [98]. Additionally, others combined both the character- and word-level features into one and utilized it for representation [91]. Although n-gram provides the probability of encountering a specific word in the text, this can fall short when having words that frequently appear in the text but do not hold significant meaning, such as stopwords that can help classify dialects. In the literature this is mitigated by adopting Term Frequency — Inverse Document Frequency (TF-IDF) in many of the works on Arabic dialects to help the classifier identify the dialects better [92], [91], [98], [94], [93], [100].

Adopting n-gram models suffers from the sparsity issue where it produces a sparse representation of, for example, a 50,000-dimensional vector where many of the values are zero-valued [53]. In the literature, several models have been adopted that produce dense vectors with a size that can be around a 300-dimensional vector [53]. These methods include word2vec, fasttext, and one hot encoding [99], [101], [102]. Although dense vectors usually allow the classifier to learn fewer weights, helping the model to generalize better [53], adopting it to learn Arabic dialect features did not yield better results than n-gram models or even yielded lower results [103], [96].

The shortcoming of the utilized static features could result from their inability to represent polysemous words in a language where each word has a fixed representation. This led to the introduction of another approach for text representation that is *con-*

textual representations, which address the shortcomings of the fixed representation in traditional approaches [104]. For example, the word *mouse* in the English language can have two meanings, the animal and the computer mouse; through providing the contextualized representation for such a word in a language can lead to better performance in sentiment analysis, question answering and named entity recognition [29].

Recent work adopting this approach showed an improvement in the ADI task since many words are shared among Arabic varieties; however, the meaning differs [65], [25]. Some work adopted mBERT (multilingual BERT) trained on 100 languages, including MSA Arabic [105], [106]. Other work utilized AraBERT, a monolingual model that is trained on MSA in [95], [106]. Finally, other work utilized ArabicBERT, which is also a monolingual model trained on MSA [96], [95]. All studies that utilized BERT-based models yielded more promising results than other statistical-based representations [105], [95]. Additionally, AraBERT outperformed mBERT in sentiment analysis and question-answering tasks [107]. Also, further pretraining Arabic on dialectal tweets produced Multi-dialectal-BERT that yielded better results than AraBERT in the ADI task[96].

3.2.3 Identification Method

The majority of works on the dialect identification task have utilized machine-learning-based classifiers as shown in Table 3.2. Although these models yield low performance in general, this can be due to the mapping of text to low-dimensional continuous feature vector with these models. The most widely used models are Naive Bayes (NB)-based models [92], [91], [98], [97], [100], [101], [108]. Also, others models are used such as

Support Vector Machine (SVM), Random Forest (RF), Logistic Regression (LR) and K-Nearest Neighbour (KNN) [98], [109], [91], [97].

Neural-based models are used for dialect classification to avoid the drawback of machine-learning-based models. These models are either simple Multi-Layer Perceptron (MLP) models utilized by few studies [108], [91], [109] or deep learning models that include Convolutional Neural Network (CNN) and RNN with its variations such as Long Short-Term Memory (LSTM), Bidirectional LSTM (BiLSTM) and Gated Recurrent Units (GRUs). These models have been utilized in various studies [99], [97], [101], [102], [105].

The majority of studies working on the task have experimented with ensemble learning in order to boost the results, and most of the results indicated the effectiveness of ensemble learning in the dialect identification task [96], [94], [95], [108], [100], [97]. Other works have shown promising results by using pre-trained language models that provide contextual representation and fine-tuning it on the classification task for dialect identification [105], [94], [95], [93].

3.2.4 Evaluation Metrics

The most common metrics in text classification are F1-score, precision, and recall [110]. The majority of work on Arabic dialect identification task uses F1-score as shown in Table 3.2 as an evaluation metric [96], [94], [95], [108], [100], [97], [105], [93]. Others have also used precision and recall [97], [97], [102], [94]. And a few others have used accuracy [92], [91], [98], [100].

Paper	Arabic Verity	Dialect Level	Feature Representation	Methods/Models			Metrics
				ML	Neural Nets	Other	
[92]	25 dialects & MSA	Region City	Ngram TF-IDF	NB	-	-	Accuracy
[91]	25 dialects & MSA	Region City	Ngram TF-IDF	SVM LR NB	MLP	-	Accuracy Precision Recall F1- Score
[97]	25 dialects & MSA	Region City	TF-IDF	NB RF	BiLSTM GRU	-	Precision Recall F1- Score
[98]	25 dialects & MSA	Region City	Ngram TF-IDF	SVM NB	-	-	Accuracy F1- Score
[99]	25 dialects & MSA	Region City	Ngram Word2Vec	NB	LSTM	-	F1- Score
[100]	25 dialects & MSA	Region City	Ngram TF-IDF	NB LR SVM Ensemble	-	-	Accuracy F1- Score
[109]	25 dialects & MSA	Region City	Ngram	NB KNN	MLP RNN	-	F1- Score
[101]	25 dialects & MSA	Region City	One Hot Encoding Fast-text TF-IDF	NB	LSTM CNN	-	F1- Score
[108]	25 dialects & MSA	Region City	Ngram	NB Ensemble	Perceptron	-	Accuracy Precision Recall F1- Score
[102]	25 dialects & MSA	Region City	Word2Vec Fast-text	-	BiLSTM CNN	-	F1- Score
[105]	25 dialects & MSA	Region City	-	Ensemble	GRU	BERT	Accuracy F1- Score
[96]	21 dialects	Country	Word2Vec Fast-text	-	-	BERT	Accuracy F1- Score
[94]	21 dialects	Country	Ngram TF-IDF		-	BERT	Accuracy Precision Recall F1- Score
[95]	21 dialects	Country	Ngram TF-IDF CountVectorize	NB LR Ensemble	-	BERT AraBERT ArabicBERT	F1- Score
[93]	18 dialects	Country	Ngram TF-IDF	SVM	-	mBERT ArabicBERT	F1- Score

Table 3.2: Summary of the recent work on Arabic dialect identification.

3.3 Arabic Neural Language Modeling

This section covers three main topics: work on word embeddings in general, non-contextualized representation for Arabic, and contextualized representation for Arabic.

3.3.1 Neural Language Models

The first large-scale neural-net-based LM reframed the problem of representing words to vector as an unsupervised learning problem [111], [112]. The presented solution projected the raw words onto the embedding layer and then these embeddings are fed into the other network layers [112]. Other studies followed, showing that word embeddings could be helpful for downstream tasks [113]. Afterwards, word embeddings gained attention when the word2vec model was introduced [114].

Other work was introduced as well such as GloVe and fastText [115] [116]. Although these models advanced the word representation, they lacked the context of the represented words. Then the first contextualized representation model ELMO was introduced to mitigate the issue using LSTM-based unidirectional context [65]. ELMO’s performance on downstream tasks is noticeably better due to the better word representation. After that, different models were introduced, such as ULMFit, which is LSTM-based as well [117]. After that, the BERT model was introduced as the first Transformer-based LM with bidirectional context [66]. Then other Transformer-based models followed such as RoBERTa that is a variation of the BERT model, XLNET, ALBERT and T5 [118] [119] [120] [121].

3.3.2 Non-Contextualized Representation for Arabic

After introducing word2vec for the English language, it was adopted for other languages as well to produce language-specific embeddings such as Arabic [122]. FastText followed on dialectal Arabic, trained on a small and noisy dataset to produce multi-

dialectal word embeddings [123]. Other work trained Arabic word embeddings on a large body of text collected from different resources [124]. Also, other works used social media and trained these language models on more than 250 million tweets and utilized it for different tasks such as dialect identification on the city level [125] or on sentiment analysis [103].

3.3.3 Contextualized Representation for Arabic

After contextualized embeddings were introduced, they were trained on the English language [65]. Then Multilingual BERT (mBERT) was trained on multiple languages (+100), including Arabic, and it performed well in most languages [66]. Other work followed where BERT was fully trained on specific languages. The first work that trained BERT on Arabic was AraBERT [107]. AraBERT was trained on MSA with different news datasets and achieved state-of-the-art performance on most tested Arabic NLP tasks, outperforming mBERT [107]. Other work trained BERT on the Arabic portion of the OSCAR dataset that is MSA as well and then utilized it for hate speech detection on social media [126]. Fully trained contextualized embeddings on dialectal text remain an open area for investigation.

3.4 Gap Analysis

After surveying the literature, the identified research gaps are in two main categories: Arabic dialect resources and Arabic dialect identification. There is a shortage of Arabic dialect resources, specifically in the data and the annotation methods. Also,

there are two issues in Arabic dialect identification regarding feature representation and the dialect identification level. The following sections describe these gaps.

3.4.1 Arabic Dialect Resources

The first step in investigating any task is looking for the available resources. After surveying the literature on Arabic dialects in Section 3.1.2 and summarizing the work in Table 3.1, two major gaps are identified, the utilized sources for collecting data and the availability of data. Arabic dialect corpora limitations are shown in Figure 3.1 where the utilized sources are user-generated based content limiting the investigated tasks on Arabic dialect [16], [76], [81], [82], [12]. The utilized sources come mainly from Twitter, as highlighted earlier, where the collected text from Twitter is short, not exceeding 280 characters [127].

Also, other sources are used, such as YouTube comments and product reviews, which allow users to write with more characters. However, the investigated tasks remain limited as well, for example, sentiment analysis [12]. Another issue when using Twitter as a data source is its limitation on sharing the data with the research community. Twitter policy prevents researchers from sharing their data if they have a premium membership and even prevents sharing the tweet’s text directly. It only allows sharing the *Tweet ID* that can be utilized to recollect the tweets. The major problem here is that many Twitter users delete their tweets or account. Hence, during the recollection process, it cannot be retrieved, affecting the reproducibility of the work [33].

The other major issue regarding data for Arabic dialects is the shortage in public

data. Additionally, the available data suffers from several issues. First, the utilized content is either domain-specific, in the travel domain, or uses seed words that lead to domain-specific content. Second, the geocode representing the dialect is based on the user-defined location in the user profile, leading to an increased number of noisy samples, as indicated by several studies [33], [10]. Hence, this work focuses on providing diverse Arabic dialect texts with a more reliable geo-location.

Another issue in Arabic dialect resources is the annotation method, as Figure 3.1 illustrates. The manual annotation requires human annotators that require allocating additional resources [14]. Additionally, a shortage of studies exploiting automatic annotation methods to label the dialect is observed. The majority of studies adopt automatic labeling by filtering the text based on the metadata, where the label for the text is based on the location tag if available [15], [16], [12]. Hence, this work provides manually annotated data and explores another automatic annotation method.

Arabic Dialect Resources				
Data Sources	Available Data		Annotation Method	
User Generated Content	MADAR	NADI	Manual	Automatic
- Limited data sources.	- Shortage of available data.		- Require resource allocation.	- Metadata.
- Specific Tasks.				
	- Domain-specific data.			
	-Inaccurate location identifier.			

Figure 3.1: Data and annotation gap.

3.4.2 Arabic Dialect Identification

Several efforts have investigated the Arabic dialect identification task. After surveying the literature on Arabic dialects discussed in Section 3.2 and summarizing the work

in Table 3.2, two main gaps have been identified. First, as Figure 3.2 illustrates, most of the adopted feature representation approaches are statistical-based, which does not capture the word context and produces sparse representations that require more computation without learning additional information [92], [91], [98], [94], [100]. Also, it requires extensive experimentation on both the character- and word-level and combining these multiple representations to obtain a suitable one.

In one particular study, the number of these LMs reaches up to 96 LMs to represent Arabic dialects [100]. Few studies utilized contextual language models for feature representation that showed better performance on the ADI task. However, the publicly available contextual models are either trained on multiple languages, including Arabic or MSA [66], [128], [107]. Hence, this work focuses on developing contextual features for Arabic dialects.

The second gap is illustrated in Figure 3.2 where the level of granularity in identifying Arabic dialects poses a challenge [90], [92], [92], [101]. Arabic dialects are mainly investigated on three levels: region, country, and city. Each level has its corresponding challenges.

On the regional level, various studies consider it inaccurate to group several linguistically different dialects [17], [6]. Meanwhile, working on the city-level dialect yields poor performance due to the high similarity among the captured dialects, making it challenging for the models to classify them [98], [100]. This motivated many studies to investigate the dialect on the country level. The challenge here lies in the collection method for this level, where the country-level dialect is represented by collecting the dialect of one or more cities in a country and grouping them to represent the coun-

try level, limiting the variation in the utilized sample. Hence, this work focuses on identifying the dialects at the country level based on the country data.

Arabic Dialect Identification				
Feature Representation		Dialect Identification Level		
Statistical	Contextual	Region	Country	City
- Lack of word context.	- Multilingual.	- Linguistic Variation.	- Grouping Cities.	- Low Performance.
	- Modern Standard Arabic.			

Figure 3.2: Representation and dialect identification gap.

CHAPTER 4

BUILDING ARABIC DIALECT RESOURCES

This chapter covers the processes for building different Arabic dialect resources. These resources include a dialectal short text corpus, manual and automatic annotated corpora and dialectal dictionaries. The process consists of data collection, preprocessing, annotation, and building the resources.

4.1 Motivation

There is a shortage of available Arabic dialect resources, as the gap analysis depicts in Section 3.4. Arabic dialects are not available in traditional media sources such as books and news articles, limiting the number of resources for Arabic dialects. However, it is present in mainstream social media such as Twitter, Facebook, and Instagram [92]. Many studies utilize Twitter to collect data, and only a few share the collected datasets for the ADI task with the research community [33]. Hence, this work uses

Twitter to collect data from different Arab countries to investigate the ADI task. Applications such as Twitter support data collection for research by providing an official API. Additionally, many Arab countries are among the top 20 countries that use Twitter to provide a rich source of informal dialectal Arabic [80].

It is essential to have annotated data to classify different dialects. After collecting the data, data annotation is vital in any classification process. It requires allocating additional resources to conduct any annotation campaign. In Arabic dialects, there is consensus among studies on the shortage of annotated corpora. To the best of our knowledge, no studies investigated automatic annotation for Arabic dialects. Hence, this work develops an automatic annotation approach that is evaluated by manually inspecting a random sample of the annotated data. It is utilized in the ADI task in a weakly-supervised learning setting and compared with the other adopted learning paradigms.

4.2 Arabic Dialect Resources

A data shortage is identified in the gap analysis phase; hence, there is a need to develop different Arabic resources to utilize at different stages in this work. Developing contextual representation requires having a large amount of unlabelled data; hence, the *Arabic Dialect Short Text Corpus* is constructed to utilize it in the pre-training of different dialectal LMs. Also, the ADI task is a classification task that requires annotated data; hence, manual and automatic annotation methods are followed to construct two Arabic dialectal corpora. A dialectal dictionary is built in the automatic

annotation phase; Figure 4.1 illustrates the final dialect resources built for each dialect:

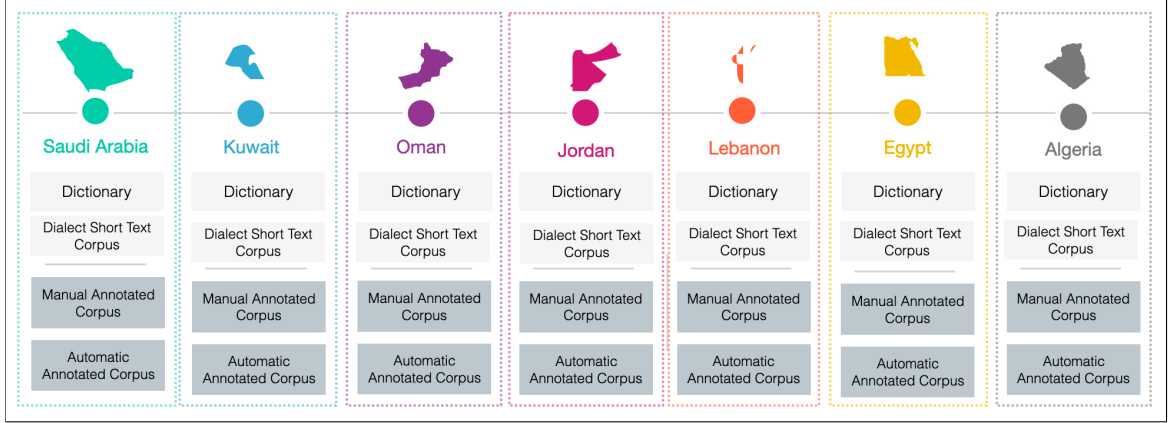


Figure 4.1: Arabic dialect resources.

1. **Arabic Dialect Short Text Corpus:** Each dialect has a large number of tweets collected with a Geo-tag as described in Section 4.3.1. Table 4.7 shows the size of the corpus.
2. **Dialect Dictionary:** Each dialect has a dictionary where that is collected, preprocessed, and augmented in a unified dictionary, as depicted in Section 4.7.1.
3. **Manually Annotated Corpus:** Each dialect has human-annotated data augmented with data from another resource human annotated as described in Section 4.8.
4. **Automatically Annotated Data:** Each dialect has data automatically annotated using the unified dictionary described in Section 4.11.

The process of building any corpora involves stages from the starting point to the resulting final version of these corpora. In this work, the process of creating different

Arabic dialect corpora is composed of four main phases, as shown in Figure 4.2. These phases are summarized as follows:

1. **Text Collection:** This phase includes collecting and archiving data from different resources as described in Section 4.3 and 4.4.
2. **Text Preprocessing:** After collecting data from different resources, especially social media, the data contains noise which requires data cleaning and text normalization to ensure the quality of the text is maintained, as described in Section 4.5.
3. **Text Annotation:** Classification of dialects requires annotated data. This work experiments with two different annotation schemes that are reflected in the data annotation.
4. **Building Annotated Corpus:** The last phase is building the final annotated corpus by balancing the data and adding the negative sample.

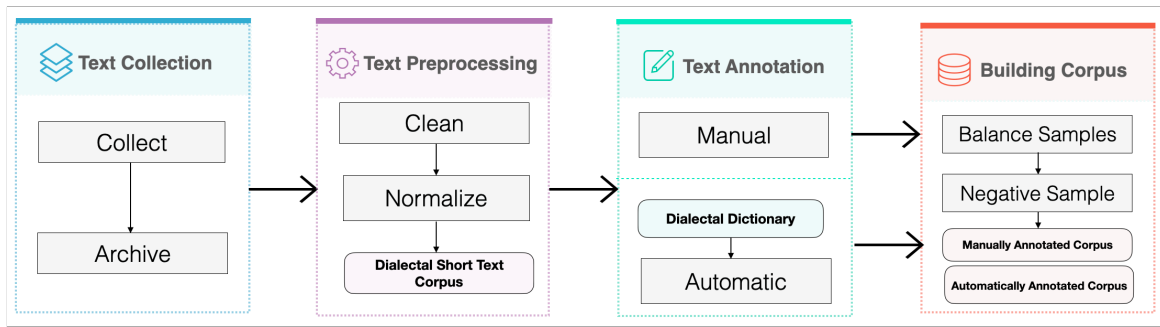


Figure 4.2: The pipeline of building an annotated Arabic dialect corpus.

4.3 Dialectal Short-Text Corpus Collection

The Dialectal Short-Text Corpus (DSTC) is collected from the social media platform *Twitter* using the official Twitter API that permits collecting data for research purposes [80]. Researchers can retrieve tweets around a specific keyword that is tweeted during the previous week only [80]. A request is raised to Twitter to justify the purpose of the research to provide the authentication key to connect to Twitter and collect data. After Twitter’s approval, Tweepy API [129] is used to connect and collect tweets. The following section provides the proposed pipeline for collecting Twitter data. Twitter allows defining different parameters in the data collection query. These parameters include the following:

- **Language:** tweets written in the specified language.
- **Country:** tweets for users that allow location sharing with Twitter.
- **Keywords:** tweets that include these keywords.

The *language* parameter retrieves tweets in the defined language, that is Arabic in our case. The *country* parameter allows retrieving tweets from a specific country for accounts that enable sharing their location with Twitter. However, only around 1% of Twitter users allow Twitter to identify their geographical location [80]. Additionally, utilizing the location tag provided by the users introduces noise, resulting in a significant challenge in collecting tweets from the targeted countries. Hence, a geographical coordinate component is proposed in our system.

Selecting keywords is a challenging task. Populations of different countries have different interests, and defining a specific set of keywords for all countries tends to retrieve a high volume of entries from one particular country more than others. Additionally, it leads to dialect-specific content rather than diversified dialect content. Hence, we propose a trending keyword module to mitigate these issues, as depicted in Figure 4.3.

The Twitter data collection pipeline is a novel language- and task-agnostic pipeline for creating a large-scale collection of tweets that can support different tasks. This pipeline can be utilized for other languages, countries, and keywords to collect tweets from Twitter. The adopted approach is demonstrated over Arabic dialect tweets from specific Arabic countries using dynamic keywords that are updated daily to ensure collecting diverse content for representing the dialect. Also, the geo-coordinate component ensures the collection of data from the specified country as identified by Twitter. The Twitter pipeline is composed of three modules: Twitter Data Collection, Geographical Coordinates, and Twitter Trends, as shown in Figure 4.3. The following sections provide the implementation details for these components.

4.3.1 Twitter Data Collection

This module retrieves the tweets and consists of six different components, as follows:

1. **Twitter Authentication:** Twitter requires passing the consumer key and the secret key to establish a secure connection and retrieve tweets [80]. Twitter provides both keys after raising a research application and getting approval.

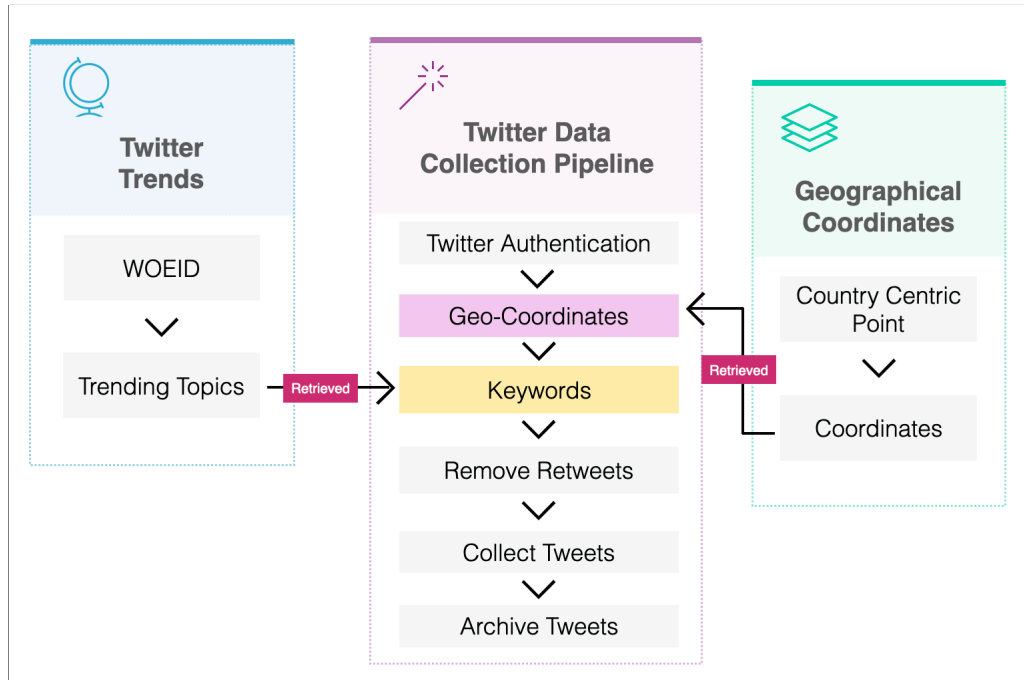


Figure 4.3: The developed Twitter data collection pipeline

2. **Geo-Coordinates:** The *Geographical* module in Section 4.3.2 passes the values for this component to use in the pipeline.
3. **Keywords:** The *Twitter Trends* module in Section 4.3.3 passes the values for this component for specific countries in a daily manner to use in the pipeline.
4. **Remove Retweets:** After establishing all the previous parameters, tweets can be retrieved. However, these tweets can contain retweeted tweets, that is, a re-posted tweet by another user, leading to data duplication where multiple tweets are identical and redundant. Removing retweets by defining that in the query mitigates this issue.
5. **Collect Tweets:** After defining the previous parameters, the query retrieves the tweets accordingly.

6. **Archive Tweets:** After retrieving the tweets, they are archived in *csv* file format in a local device.

4.3.2 Geographical Coordinate Data Collection

This component provides the countries' coordinates for the *Twitter Data Collection* module. It consists of two parts as follows:

1. **Country-Centric Point:** To ensure collecting dialectal tweets from a specific country, one of the parameters that a Twitter query can take is the geographical *latitude* and *longitude* that defines a specific location on the map to collect tweets from users who tweet in that area [3]. An online tool obtains these coordinates by specifying a specific geographic centering point on the country's map, as Figure 4.4 illustrates. Also, the tool provides the circle radius value around the centering point that allows for curating all the tweets in the circle radius inside each country.
2. **Coordinates:** After retrieving the countries' coordinates along with the area of the circle radius as Table 4.1 shows, another online tool is used to verify the coordinates, and it yields identical results [130].

4.3.3 Twitter Trends Data Collection

This module retrieves the trending words for each country and passes them to the Twitter Data Collection module. It utilizes these trending words as keywords for collecting tweets. This ensures covering diverse topics that are circulating among the



Figure 4.4: Specifying a Centering Geographical Point in Saudi Arabia Using Free map Online Tool [3].

Country	Latitude	Longitude	Circle Radius
Saudi Arabia (SA)	23.4869	44.8203	614 Km
Egypt (EG)	27.6400	29.1831	404km
Algeria (DZ)	33.7770	3.2066	417 Km
Kuwait (KU)	29.4362	47.6939	70 Km
Oman (OM)	22.3167	57.6578	214 Km
Jordan (JO)	31.4168	36.2754	82 Km
Lebanon (LB)	33.8149	35.6607	29.93 Km

Table 4.1: Country coordinates.

population of each country and consequently having a dataset that covers diverse topics in each dialect. This module has two components as follows:

1. **Where On Earth ID (WOEID):** To retrieve the trending words from each country, Twitter requires passing WOEID for each country to provide those keywords. Unfortunately, the WOEID is not available for all Arab countries. Only a few countries possess an active WOEID. These countries are from four different geographical regions in the Arab world. Table 4.2 shows the WOEID for each country.
2. **Trending Words:** After defining the WOEID for each country, the trending

Country	WOEID
SA	23424938
EG	23424802
KU	23424870
JO	23424860
LB	23424873
OM	23424898
DZ	23424740

Table 4.2: Country WOEID [1].

words are retrieved, and Twitter can retrieve up to 50 words per day. However, it varies from country to country and from one day to another. For example, countries with many users can produce a more considerable number of trending words since different users have different interests. Additionally, if an event occurs, such as an international football game, the volume of trending words increases, and if no noticeable events occur, the number of trending keywords decreases. Table 4.3 shows the average number of retrieved keywords per country.

After running the pipeline and collecting tweets from Twitter, a variation in the volume of tweets is encountered where some countries could retrieve 30,000 tweets per day. This can take more than six hours on average due to Twitter’s time limitation during data collection. Hence, four different authentication keys are used to collect the tweets for around seven months, from 15-09-2020 to 10-03-2021. The average statistics in Table 4.3 show the retrieved keywords, number of tweets, and time duration required for the pipeline to keep running until finishing the tweet collection per day.

Country	Keywords	Tweets	Time (Minutes)
Saudi Arabia	49	21,165	343
Egypt	47	25,169	360
Kuwait	39	20,418	240
Jordan	19	3,318	53
Lebanon	30	3,356	56
Oman	24	3,418	155
Algeria	17	1,442	45

Table 4.3: Average daily statistics for running the proposed Twitter pipeline.

4.3.4 Data Evaluation

In order to ensure the effectiveness of the pipeline in capturing dialectal content from each country, 100 random samples from each country are reviewed manually. This process is repeated three times on a new random sample each time. The dialect is present in these samples with varying degrees, as Table 4.4 shows in the average results for each dialect. The highest percentages are in Saudi Arabia and Egypt, which have the largest amount of data among other dialects. This is possibly due to the number of active users in each country, as seen in Table 4.3. Other dialects have fewer dialect samples compared to Saudi Arabia and Egypt. Additionally, this can be due to the volume of the data retrieved each day, indicating less engagement from users in these countries. Also, the presence of MSA is prominent in the tweets in the form of prayers and news since both are usually written in formal Arabic or users writing in MSA, adding noise to the data. Other minor samples also belonged to another dialect rather than the original dialect.

Country	#Dialectal
EG	62%
SA	49%
KU	37%
LB	37%
DZ	35%
OM	27%
JO	19%

Table 4.4: Data validation for the collected data from the proposed pipeline.

4.4 Dialect Dictionary Collection

A dictionary is a list of words and symbols used to automatically label data in case human annotation is unavailable, as it is a cost-effective method [53]. In this work, different Arabic dialectal dictionaries are built from different resources. The following sections provide the implementation details.

4.4.1 Crowd-sourced Dialect Dictionary

This dictionary uses an online source that contains a set of dialectal words for data collection; it is the largest online source that archives dialectal Arabic words, phrases, and proverbs for all Arabic dialects [131]. This source allows online users to add new words to the dictionary of each dialect. Additionally, the users can evaluate the previous content by other users by giving *like* if the word belongs to the dialect or *dislike* otherwise. The curated data combines words and phrases to build the dictionary for each dialect. The phrases were split into words and added to the dictionary; redundant words were removed. Table 4.5 shows each dialect’s vocabulary size.

Country	Vocabulary Size
SA	4297
DZ	3149
EG	1477
JO	784
LB	748
KU	611
OM	451

Table 4.5: Crowd-sourced dialectal dictionary.

4.4.2 Curated Dialect Dictionary

The curated data is from the Atlas website [132]. This website introduces words and their synonyms in other Arabic dialects and their origins, whether they are Arabic or non-Arabic. The resources for building this dictionary include Arabic books written about the Arabic language and its dialects; several linguists also reviewed the dialectal content. Table 4.6 shows the size of the vocabulary for each dialect.

Country	Vocabulary Size
SA	3035
DZ	896
EG	962
JO	742
LB	638
KU	1414
OM	1147

Table 4.6: Curated dialectal dictionary.

4.4.3 Arabic General Purpose Lexicon (AGPL)

Looking at any text in books, documents, articles, and social media posts, it is clear that not all the words carry significant meaning. One kind of common words that do not convey any meaningful information is stopwords. Removing stopwords from

the dialect dictionary ensures having only dialectal words. We build different lexicons by compiling different online resources that provide Arabic stopwords, and several datasets are obtained. However, these datasets have a varying number of Arabic stopwords. One study presented a comprehensive list of Arabic stopwords embedded in the author’s website [133], [134]. The website is scraped using Python scrapy package to build the AGPL list locally. This resulted in building an AGPL list that consists of **1,614** stopwords.

4.5 Data Preprocessing

Although social media is a valuable resource for collecting data, user-generated content is considered less standardized and contains noise [135]. Also, scraped data from the web includes additional irrelevant information that introduces noise to the text. This information includes the page scripting code, links, symbols, punctuation marks, and other irrelevant data points that can affect the performance of the models on the ADI task. In order to obtain clean and relevant text, several preprocessing steps have to be done. This includes the data cleaning from unwanted noise such as symbols or web script from the web page content. In addition, textual data, specifically user-generated text, requires standardization to utilize in different tasks; this is done in the normalization phase. This section describes the preprocessing tasks to obtain clean text from the available data.

4.5.1 Dialect Short-Text Corpus

In order to obtain a clean text that can help provide representative features, the data requires several preprocessing steps described in the following sections for each resource.

Data Cleaning

Most of the data used in this work are collected from Twitter, as described in Section 4.3.1. Twitter allows retrieving tweets within one week from the current day of collection. This can lead to retrieving a tweet multiple times since the data is collected daily, especially if the trending word has been trending for several days or if the specified country produces a low number of tweets daily. To remove duplicates, the daily tweets' files are combined into one file. Next, duplicate tweets are removed based on the tweet ID, where each tweet has a unique ID assigned to it from Twitter. This step causes a significant reduction in the number of tweets, as shown in Table 4.7. Moreover, to ensure capturing a representative sample of the tweets, tweets with fewer than seven tokens are removed, which affects the data size, as shown in the Table below.

Country	Retrieved Tweets	Unique Tweets	7+ Tokens Tweets	Reduction
Saudi Arabia	4,693,533	3,614,590	2,415,622	46%
Egypt	5,677,800	3,313,610	2,099,977	61%
Kuwait	4,047,308	823,546	477,973	88%
Oman	665,463	316,500	200,384	61%
Lebanon	670,715	294,275	204,430	59%
Jordan	657,472	232,124	97,400	67%
Algeria	245,480	115,564	103,488	58%

Table 4.7: Twitter dataset size and the effect of cleaning on the tweets

Text Normalization

Text normalization is the process of converting text into a more convenient and standardized form [53]. Normalization is a critical step when working on languages since having any variation, even in one letter, can treat the same word as a different word, especially with Arabic dialects, where sometimes the difference among dialects in a word is a letter that can produce inaccurate classification [136]. The noise in the text includes usernames, links, hashtags, symbols, digits, and punctuation marks, which have to be removed from text to have standardized and clean data. Text normalization enables having a standard form of the text [53]. One of the most well-known methods in NLP to extract strings from text following specific patterns is *regular expression* [53]. The rules for cleaning and normalizing text are written as regular expression rules performed on text. Several steps are followed to ensure the normalizing of text, including the following:

- Arabic letters can have different varieties; these letters were mapped to one variety. For example, the letter Alef (ا) has four varieties (أ, إ, إ, إ).
- The Arabic diacritics for the vowel marks are removed.
- Repeated Arabic characters are removed if repeated more than two times in the text.
- Arabic digits were removed.

Figure 4.5 shows the preprocessing steps described earlier. In the example, the English characters are removed, then the symbols, followed by punctuation present in

the raw tweet that is (!!!), then, the repeating characters, followed by unifying the different varieties of a letter to a certain variety.

Remove English Characters	@b78htNnyMIIUZKu @MEWA_KSA @News_Almadinah @hazmiharbi @snaptaibah1 @soalef_almadina @qweasdzc998671 @m_e_d_mujtamae @madinah_events @Trend_MD جزاك الله خييبيير!!! ياراعي الجود حبيب قلبي أبووو ياسر
Remove Symbols	جزاك الله خييبيير!!! ياراعي الجود والوفاء حبيب قلبي أبووو ياسر @ @ @ @ @ @ @ @ @
Removed Punctuation	جزاك الله خييبيير!!! ياراعي الجود والوفاء حبيب قلبي أبووو ياسر
Normalize Repeating Characters	جزاك الله خييبيير ياراعي الجود والوفاء حبيب قلبي أبووو ياسر
Unify Letters	جزاك الله خير ياراعي الجود والوفاء حبيب قلبي أبو ياسر
Final Text	جزاك الله خير ياراعي الجود والوفاء حبيب قلبي ابو ياسر

Figure 4.5: Preprocessing Steps on a Sample Tweet.

4.6 Building a Manually Annotated Corpus

After collecting, cleaning, and annotating the data, the resulting data remains unready for the dialect identification task. For the algorithm to identify if the text belongs to a specific dialect or not, it needs to learn from the data, both the positive (dialectal text) and negative (other dialects). Hence, the negative sample needs to be added to the data. After annotating the data, the dialect will be represented by a specific size of samples. Then, a number negative samples equal to the number of positive samples are

added to obtain a balanced dataset. Our proposed approach for developing annotated corpora follows these steps:

- **Annotation:** Annotate the data using the selected annotation scheme.
- **Negative Sample Generation:** Negative sampling utilizes other dialectal labeled data as a negative sample to produce balanced data in order for the algorithm to learn to distinguish between different Arabic dialects. This phase is described below.

4.6.1 Manual Annotation

Annotators annotate the tweets; each dialect is labeled by an Arabic speaker and reviewed by another reviewer. Annotators are either native speakers or knowledgeable in the dialect, especially if the annotator belongs to the same geographical region as the other dialect; for example, Saudi Arabia and Kuwait are both in the Gulf region. People from the same region can easily recognize each other's dialects.

The annotators are volunteers for the following dialects: Saudi Arabia, Kuwait, Oman, and Egypt. In other dialects, the annotators are not domain experts nor knowledgeable about it. These dialects are Algeria, Jordan, and Lebanon. Native speakers from these dialects are recruited to label the data from an online freelance service website [137].

The process of annotation is depicted in Figure 4.6. First, the unlabeled data passes to the local annotators based on their profile location. Then the annotators annotate the sample as dialectal if it contains two dialectal words or more. Then the

labeled data is reviewed by either a local reviewer or a recruited reviewer. Then the final version of the annotated data is archived.

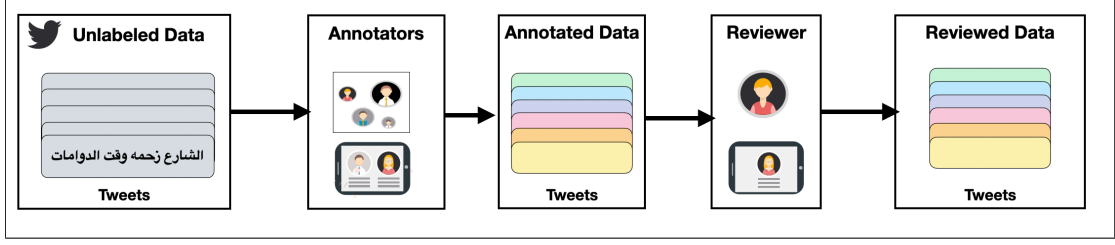


Figure 4.6: Supervised Learning Annotation Process.

Country	Our Data	NADI	Total
Saudi Arabia	1,000	1,000	2,000
Algeria	1,000	1,000	2,000
Egypt	1,000	1,000	2,000
Jordan	1,588	412	2,000
Lebanon	1,401	599	2,000
Kuwait	1,600	400	2,000
Oman	1,000	1,000	2,000
Total	8,589	5,411	14,000

Table 4.8: Manually Annotated Data.

4.6.2 Negative Sample Generation

Supervised learning models require annotated data. The previous section describes the annotation process of each dialect. In order to produce a negative sample for each dialect, the following equation is performed:

$$D_{Neg} = \sum_{n=1}^{n=6} \frac{d_n}{6}$$

The negative sample for each dialect is sampled from other human-annotated dialectal data where D_{Neg} denotes the negative portion of the corpus and d_n represents the portion of the negative sample taken from each other dialect. The sum of these small portions equals the size of the positive sample. In human-annotated data, the

number of the positive samples is 2,000 . The negative sample must be 2,000 samples to produce a balanced supervised learning corpus. The number of random samples sampled from each dialect is 333, resulting in a balanced dialect ground truth corpus of 4,000 samples as shown in Table 4.9.

Dialect	Positive	Negative							Total
		SA	DZ	EG	JO	LB	KU	OM	
SA	2,000	333	333	333	333	333	333	333	4,000
DZ	2,000	333	-	333	333	333	333	333	4,000
EG	2,000	333	333	-	333	333	333	333	4,000
JO	2,000	333	333	333	-	333	333	333	4,000
LB	2,000	333	333	333	333	-	333	333	4,000
KU	2,000	333	333	333	333	333	-	333	4,000
OM	2,000	333	333	333	333	333	333	-	4,000

Table 4.9: Negative Sample Generation for Each Dialect.

4.7 Building Automatically Annotated Corpus

Building an automatically annotated corpus follows a process similar to that of the manually annotated corpus described earlier. However, automatic annotation requires an external data source to annotate the data. An extra step is added to annotate Arabic dialects by building the dialectal dictionary.

4.7.1 Building a Dialectal Dictionary

A Dialectal Dictionary (DD) is collected from two different resources as described in Sections 4.4.1 and 4.4.2. In order to have a unified dictionary for each dialect, several preprocessing steps are followed. Figure 4.7 illustrates a general overview of the steps to build the dialectal dictionaries.

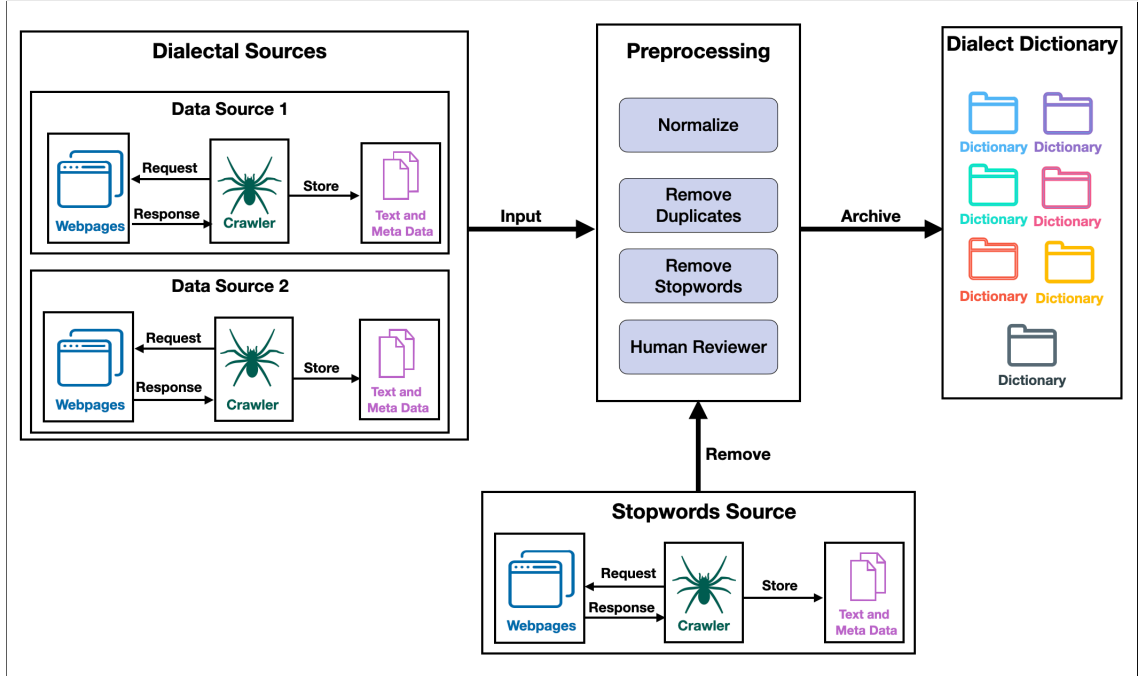


Figure 4.7: The Process of Building Arabic Dialectal Dictionaries.

To build a unified, clean dictionary for each dialect, the following steps are followed:

- **Step 1:** Both dictionaries are scraped from different resources as mentioned in Section 4.4.
- **Step 2:** All phrases are split into words and the phrase words are included in the dictionary, if the words does not already exist in the dictionary.
- **Step 3:** After collecting the dictionaries, both contained many stopwords. This led to collecting another general-purpose dictionary that contains stopwords as described in Section 4.4.3. Then, this dictionary is used to remove stopwords from both dictionaries.
- **Step 4:** Both dictionaries are cleaned manually to remove: any remaining MSA words and phrases, toxic speech, meaningless input such as symbols or arbitrary

user input, space between letters in a single word, and irrelevant input such as website code.

- **Step 5:** Both dictionaries are combined into a unified lexicon for each dialect.

Algorithm 1 describes the process of building the DD. The outcome of each step is shown in Table 4.10 for each dialect in this work. The raw crowd-sourced collected data is the first dictionary, and the curated dictionary is the second. The first dictionary contains phrases that are split, the new words are added, and redundant ones are removed. Then it is preprocessed as described in Section 4.5. Finally, the AGPL is removed, and both dictionaries are combined.

Country	Raw Mo3jam	Clean Mo3jam	Atlas	Phrase	Phrase Words	Added Words	Removed Words	AGPL	DD
SA	3,522	3,385	3,035	751	2,173	1,477	696	88	7,045
DZ	2,632	2,575	896	462	1,284	927	357	66	3,869
EG	1,145	1,041	962	220	624	488	136	41	2,227
JO	624	595	742	121	341	260	81	21	1,453
LB	480	454	638	99	260	219	41	15	1,195
KU	595	549	1,414	100	267	225	42	20	2,066
OM	351	331	1,147	59	177	145	32	12	1,550

Table 4.10: Building Dialect Dictionary Process

4.7.2 Automatic Annotation

There is a need for annotated Arabic dialect corpora in the literature. The adopted approaches for annotating the Arabic dialects are mainly manual. Another approach for annotating text is using automatic annotation that uses computational functions that can generate a label or a tag that describes some textual content [138]. Some studies on Arabic dialects adopt automatic annotation using the metadata of the retrieved text. This approach can increase the noise in the data or cause losing good

Algorithm 1: Build Dialect Dictionary Algorithm

```
1 Unify Dictionary (Mo3jam, Atlas, AGPL);  
   Input : Mo3jam Words Mo3jam,  
           Atlas Words Atlas,  
           Arabic General Purpose Lexicon AGPL  
   Output: Dialect Dictionary  $D_D$   
2  $D_D = []$ ;  
3 for  $word_i \in Atlas$  do  
4   | if  $word_i \notin D_D$  then  
5   |   |  $D_D \leftarrow word_i$ ;  
6 end  
7 for  $word_i \in Mo3jam$  do  
8   | if  $length(word_i) > 1$  then  
9   |   |  $Words = word_i.split()$ ;  
10  |   | for  $word_j \in Words$  do  
11  |   |   | if  $word_i \notin D_D$  then  
12  |   |   |   |  $D_D \leftarrow word_j$ ;  
13  |   | end  
14  | else  
15  |   | if  $word_i \notin D_D$  then  
16  |   |   |  $D_D \leftarrow word_i$  ;  
17  | end  
18 end  
19 for  $word_i \in D_D$  do  
20 | if  $word_i \in ASW$  then  
21 |   | remove  $word_i$ ;  
22 end  
23 return  $D_D$ 
```

samples that can enrich the data. This work develops an automatic annotation method for annotating the data to provide an approximate label representing the dialect cost-effectively without requiring many computational or human resources.

The automatic annotation process requires additional data sources that the developed labeling function can utilize to generate the automatic label. However, the available data resources for each dialect are limited to corpora, as seen in the investigated literature. Different dictionaries are collected from various resources as

described in Section 4.4 then cleaned and constructed to be able to perform this annotation approach. Figure 4.8 illustrates an overview of the developed automatic annotation approach to the tweets.

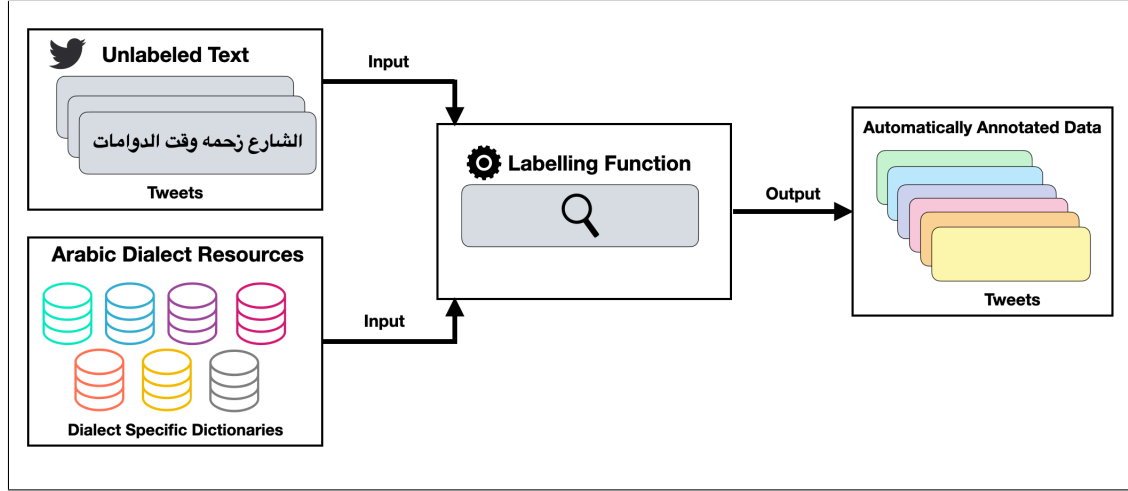


Figure 4.8: Tweets Automatic Annotation Process.

The retrieved data from each country is archived separately and the dictionaries are separated to minimize capturing a tweet that belongs to another dialect, since one of the challenges when working with Arabic dialects is the shared vocabulary, as mentioned in Section 2.1.3. The tweets are automatically annotated using the developed dialect-specific dictionary, where the tweets curated from a particular country are labeled if the tweet contains two tokens or more from the corresponding country's dictionary as Algorithm 2 depicts. Figure 4.9 provides an example tweet curated from Kuwait and labeled as being Kuwait dialect.

Tweet	Identified Words
خوش حظر والله قاعدين يلعبون هالحزه	['خوش', 'هالحزه']

Figure 4.9: Sample Tweet that is Automatically Annotated.

Algorithm 2: Automatic Annotation Algorithm.

```
1 Automatic Annotation ( $D_U$ ,  $dialect\_dict$ );  
   Input : Unlabelled data  $D_U$   
   Output: Labelled Text  $D_W = \{ D[w_1], \dots, D[w_i], \dots, D[w_m] \}$  for each  $t_i \in D_U$   
2 for  $t_i \in D_U$  do  
3   for  $word$  in  $dialect\_dict$  do  
4      $count\_dialect\_words = 0$   
5     if  $count\_dialect\_words == 2$  then  
6        $t_i.label = word.label$ ;  
7        $D_W \leftarrow t_i$ ;  
8     else if  $word \in t_i$  then  
9        $count\_word += 1$ ;  
10  end  
11 end  
12 return  $D_W$ 
```

After labeling the data, the size of the data is reduced significantly by a total of 72% across all dialects, as Table 4.11 shows. The reduction can be attributed to the proportional size of the data and the number of words in the country's dictionary. For example, Algeria has the smallest data size among all dialects. However, the amount of data reduction is among the lowest since the Algerian dialect dictionary is larger than most of the other dialects, allowing more dialectal data to be captured.

Country	Retrieved	Annotated	Reduction %
SA	2,544,678	912,152	64%
DZ	103,487	30,930	70%
EG	2,208,098	655,891	70%
JO	171,300	8,748	95%
LB	215,080	10,155	95%
KU	501,237	26,530	95%
OM	215,037	14,834	93%
Total	5,958,922	1,659,240	72%

Table 4.11: Automatically Annotated Data Size.

Data Evaluation

In order to evaluate the effectiveness of the proposed weak annotation approach, a random sample of 100 samples is drawn in three iterations. In case of an overlap between iterations, the duplicate samples are replaced. Next, these samples are evaluated manually as either actual dialectal samples or not. The results of the evaluation vary between dialects as shown in Table 4.12. The average percentage refers to the captured true dialect in the three iterations. For example, the true dialectal samples in the EG dialect are 81/100, 82/100, and 88/100, respectively. The results are averaged on the number of iterations and then reported. Observing the noise in the weakly annotated samples shows that most of these samples are MSA, specifically, religious prayers, as Figure 4.10 shows. Similar samples are present across dialects, in addition to news tweets, ads, and users who prefer writing in MSA.

Country	#Dialectal
LB	93 %
EG	84 %
JO	69 %
SA	65 %
KU	65 %
DZ	43 %
OM	36 %

Table 4.12: The Evaluation of the Weak Annotation Approach.

4.7.3 Negative Sample Generation

An approach similar to that used in the supervised negative sampling is adopted. However, neither *Saudi* nor *Egypt* dialects can be uniformly sampled from other dialects without sacrificing the uniform distribution of the negative sampling. Hence, a

اللهم صل وسلم علي اشرف الخلق نبيك محمد صلي الله عليه وسلم يوم الجمعة	SA
صباح الخير يوم الجمعة اكثروا من الصلاة علي نبينا محمد وعلي اله وصحبه اجمعين وسلم تسليما كثيرا مباركا فيه	OM
بسم الله الذي لا يضر مع اسمه شيء في الارض و لا في السماء و هو السميع العليم صباح لخير	KU
مع نسائم الصباح الجميل اسأل الله ان يرزقكم فرجه لا يأتي بعدها حزن ولا هم وسعاده لا يأتي بعدها شقاء ورزق لا تأتي بعده حاجه الاردن اليوم	JO
سبحان الله الحمد لله لا اله الا الله الله اكبر لاحول ولاقوه الا بالله اللهم صل وسلم علي محمد و علي ال محمد سبحان الله ويحمده سبحان الله العظيم استغفر الله واتوب اليه استغفر الله	EG

Figure 4.10: Samples from the Weak Positive Data for Different Dialects.

number equal to the smallest annotated dialect that is *Jordan* is randomly sampled from other dialects producing a 52,488 negative sample size. As a result, a random sample of positive tweets were taken to represent both the *Saudi* and the *Egypt* dialects as Table 4.13 shows.

Dialect	Positive	Negative							Total
		KSA	DZ	EGY	JO	LB	KU	OM	
SA	52,488	-	8,748	8,748	8,748	8,748	8,748	8,748	104,976
DZ	30,930	5,155	-	5,155	5,155	5,155	5,155	5,155	61,860
EG	52,488	8,748	8,748	-	8,748	8,748	8,748	8,748	104,976
JO	8,748	1,458	1,458	1,458	-	1,458	1,458	1,458	17,496
LB	10,155	1,692	1,692	1,692	1,692	-	1,692	1,692	20,304
KU	26,530	4,421	4,421	4,421	4,421	4,421	-	4,421	53,052
OM	14,834	2,472	2,472	2,472	2,472	2,472	2,472	-	29,664

Table 4.13: Negative Sample Generation for each Dialect.

CHAPTER 5

ARABIC CONTEXTUAL REPRESENTATION

This chapter covers the mono-dialectal Arabic contextual language representation models.

5.1 Motivation

The work on contextual LMs, also known as Pre-trained Language Models (PLMs), shows their ability to learn universal language representation. That enables NLP downstream tasks to perform better without allocating substantial resources to train these models from scratch for each task [67]. These models are pretrained on large unannotated text datasets requiring fewer annotated samples for downstream tasks. These models are suitable for the ADI task since there is a shortage of publicly available annotated datasets. Embeddings capture the semantic meaning of the words. However, the few initial works failed to capture the context and higher-level con-

cepts such as polysemous disambiguation, syntactic structures, semantic roles, and anaphora [67]. Many Arabic words are shared among dialects; however, these words could hold a different meaning in each dialect. On the other hand, the contextual PLMs actually focus on contextual word representation such as ELMo [65], OpenAI GPT [139], BERT [66], and RoBERTa [118].

Although these PLMs are trained on large datasets and are made publicly available and integrated into different systems, they fail to support all languages, specifically Arabic dialects. Certain studies indicated that using PLMs can enhance the performance of the ADI task [105], [94], [95]. Accordingly, the contextual language model is utilized to represent each dialect in this work. This chapter describes developing Arabic PLMs for the different Arabic varieties investigated in this work. The selected model is a variation of the BERT model [66], RoBERTa [118]. It introduces several improvements over the BERT model, as described below, where it outperformed BERT in several downstream tasks.

5.2 Transformer-Based Language Models

Robustly Optimized BERT Pre-training Approach (RoBERTa) is a variation of BERT, the Transformer-based model described in Section 2.3.4. The model is developed as an optimization of BERT; however, as the name suggests, it optimizes the BERT model by changing a few components and hyperparameters. The differences between BERT and RoBERTa are described in the following subsections as the main BERT model is discussed in Section 2.3.4.

RoBERTa is a replication study of BERT, modifying over BERT to provide more improvement[118]. These modifications include the following:

- **Text Tokenization:** RoBERTa utilizes Byte-per-Encoding (BPE), whereas BERT uses WordPiece as described in Section 2.3.4.
- **Masking:** The MLM approach in RoBERTa is performed dynamically during the training process, whereas BERT performs it statically once before the pre-training as mentioned in Section 2.3.4.
- **Training Objective:** The NSP in BERT is removed from RoBERTa.
- **Batch Size:** RoBERTa is trained on a larger batch size compared to BERT.

In the following sections, a detailed description of these differences is covered.

Text Tokenization

The text tokenization utilized by RoBERTa is Byte-Pair Encoding (BPE) [118]. The BPE is a hybrid method between the word- and character-level representation, allowing for larger common vocabularies to be derived from the corpora [118]. BPE works on subword units that are extracted from the training corpora using a statistical analysis method [118]. The usual vocabulary size is between 10k and 100k subword units, whereas BPE implementations utilize *bytes* rather than Unicode subword characters as the base for the subword units. It produces a smaller vocabulary size of 50k subword units [118], [140], encoding any text input without leaving *unknown* tokens or what is known as *UNK* tokens [118], [140]. The produced vocabulary adds approximately

15M to 20M additional parameters to the BERT without additional preprocessing or tokenization, making it a better option for encoding text [118].

Traditional tokenization in NLP splits the sentence based on the blank space, such as *'low lower lowest'* is split to *['low', 'lower', 'lowest']* [141]. However, the relationship between these words remains unknown to the model. For example, suppose the words 'new', 'newer', and 'newest' are introduced. In that case, the semantic meaning is not captured by the model [141]. BPE is proposed to mitigate this issue.

In order to understand BPE, let us take the example proposed in the original paper shown in Figure 5.1 [140]. There are several tokens and their corresponding frequency in the corpus. At the end of each token, a special stop token is marked by adding *</w>*[140]. Next, each token is split into characters as shown in the figure: *'low </w>'* is split into *['l', 'o', 'w', '</w>']* [140], [141]. At each iteration, each consecutive byte pair is counted, and the most frequent pair is merged into a single token as shown in Figure 5.1. The byte pair *'e'* and *'s'* frequency were 3+6=9 that corresponds to the most frequent and merged into *'es'*[141]. This is repeated similarly across the example, until the number of tokens or the number of iterations is defined [140].

Masking

BERT performs random masking after the tokenization process and before feeding the text into the embedding layers once, as discussed in Section 2.3.4 [71], [118]. This results in single static masking of the tokens, which yields lower accuracy compared to dynamic masking utilized by RoBERTa [118]. The dynamic masking is performed



Figure 5.1: BPE text processing example.

during the pre-training phase at each epoch, where the training data is duplicated ten times and then masked [118]. At each time, the masking is performed differently on these input sequences, which results in performing the masking in ten different ways on the training instances [118].

Training Objective

The BERT model has the NSP training objective as described in Section 2.3.4. However, RoBERTa removes the NSP training objective and performs several empirical experiments to evaluate the performance of the model with and without the NSP objective [118]. The results showed that the performance of the model after discarding the NSP matches the performance of using it and in some cases, it outperforms it [118].

Batch Size

Inspired by several studies indicating that training BERT on a larger batch size can yield better results [142], [143], the RoBERTa model adopted the larger batch size during the training process, yielding better results for the masked language modeling objective and better accuracy on the downstream tasks [118]. Training on larger batches is also computationally efficient since the data can be parallelized by distributing it through parallel training [118].

5.3 Arabic RoBERTa (AraRoBERTa)

Developing a contextual language model requires allocating different resources and performing several steps before having a developed language model representation. Prior to providing the dialectal text to RoBERTa to produce dialectal LMs, namely, AraRoBERTa, dialect tokens are properly tokenized by developing the BPE for each dialect. After collecting, preprocessing, and encoding the data, it is fed to RoBERTa to develop the contextual representation LM for each dialect. After developing the LMs, each LM is saved to be utilized in the fine-tuning phase for the text classification task.

The dialectal PLMs are then fine-tuned on the classification task, specifically dialect classification, also known as dialect identification. Figure 5.2 illustrates the different phases of developing the Arabic dialect LM, namely, AraRoBERTa variations. The pre-training phase requires large text, and the fine-tuning phase is supervised training on the classification task on the labeled dataset described in the following

chapter. The following sections describe the AraRoBERTa training approach, the implementation, training procedure, and results.

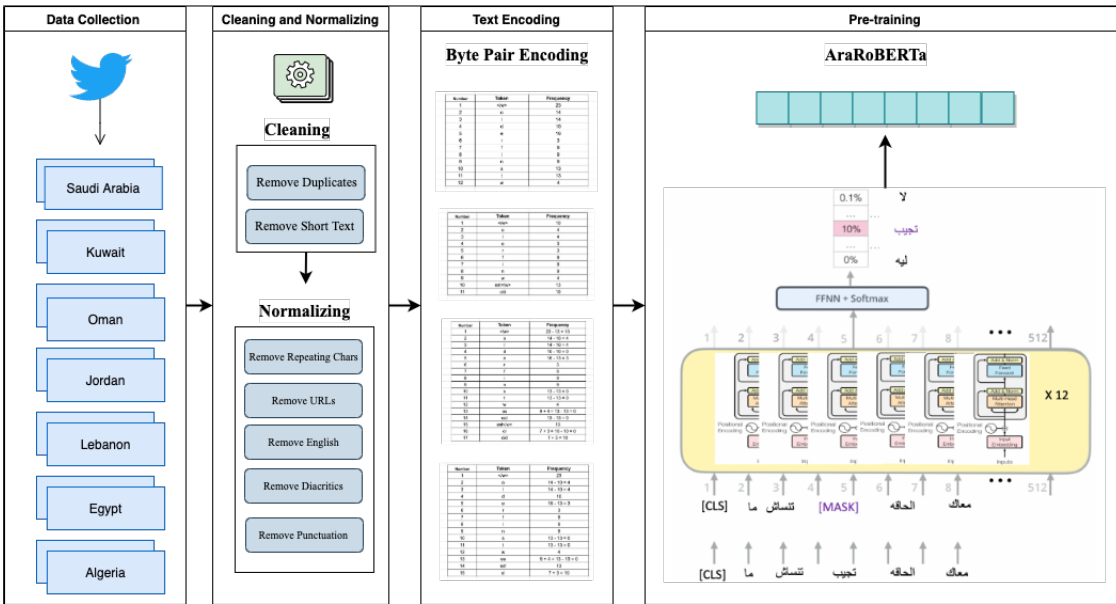


Figure 5.2: The process of developing AraRoBERTa.

5.3.1 Training Data

AraRoBERTa varieties are trained on different Arabic datasets, each on a certain dialect. Table 5.1 shows the description of each dataset: the dialect and the corresponding pretrained model, the number of tokens, the dialect vocabulary, and the percentage of the vocabulary to the tokens. More details are described in the data collection Section 4.3 and data preprocessing Section 4.5.

5.3.2 Tokenizers for Arabic Dialects

The AraRoBERTa models utilize BPE-based tokenizers to tokenize text to subwords since the RoBERTa model uses a BPE tokenizer [118]. Each AraRoBERTa model has

Dialect	PLM	Tokens	Vocabulary
SA	AraRoBERTa-SA	45,404,533	1,120,615
EG	AraRoBERTa-EG	37,273,512	944,448
KU	AraRoBERTa-KU	8,938,337	59,404
OM	AraRoBERTa-OM	3,888,023	49,692
LB	AraRoBERTa-LB	3,576,639	223,546
JO	AraRoBERTa-JO	2,632,853	171,666
DZ	AraRoBERTa-DZ	1,881,611	180,881
All Dialects	AraRoBERTa-All	103,595,508	3,250,252

Table 5.1: AraRoBERTa pre-training data.

a corresponding tokenizer that is trained on the same training data as the model, as Table 5.1 shows. The termination criteria for the tokenizer is reaching the specified number of vocabulary that in our work is 52,000 tokens. The number of tokens is empirically defined after evaluating the BPE tokenizer on 100 languages, including Arabic, by the original work [140].

5.3.3 AraRoBERTa-XX Models

The AraRoBERTa models shown in Table 5.1 are built using HuggingFace’s Transformers API [144]. The API is an open-source library that provides the implementation and various state-of-the-art pre-trained models, e.g., BERT. The research community can openly use and share these models, supporting the use of this package by researchers, practitioners, and industrial deployment [144]. It provides the tools to pre-train and fine-tune large models.

In order to prepare the raw data for pretraining, the collected datasets are pre-processed in two stages as described in Section 4.5. The raw entries are cleaned from irrelevant data points, and the acquired text is normalized. Finally, the data is tokenized via the BPE tokenizer described in Section 5.2. The BPE resolves the OOV

problem and provides a smaller vocabulary size, making it more efficient [140].

In this work, Google Colaboratory GPU (1x16GB NVIDIA Tesla P100) is utilized for the model pretraining. The AraRoBERTa models are based on the RoBERTa model which is a variation of the BERT model [118]. The architecture is RoBERTa-based with MLM pre-training objective configuration that is based on the BERT-based configuration[118]. It consists of 12 encoder layers/blocks, 768 hidden dimensions, 12 attention heads, 512 maximum sequence lengths, and of 125M parameters [66], [144]. Table 5.2 depicts the model configuration. The batch size can vary based on the performance and the available computational power [66], [118]. In this work, the selected batch size is 32 due to the limited computational resources since larger batch size require larger memory size.

Configuration	Value
Vocabulary Size	52,000
Encodes	12
Attention Heads	12
Hidden Size	768
Intermediate Hidden Size	3,072
Activation Function	GELU
Dropout	0.1
Maximum Sequence Length	512
Batch Size	32

Table 5.2: AraRoBERTa Pre-training Configuration

In initial experiments, the models are trained on three epochs. However, the performance of these models did not yield satisfactory results, indicating the models require further training. Hence, the number of epochs is increased similar to other works that adopt similar approach [66], [118], [107]. The models are trained on mul-

multiple runs due to computational limitation that limits the duration of training for users. Where the utilized resources are the max settings from Google Celebratory for premium users. On each run, the model is trained on a specific number of epochs, and then the model checkpoint is loaded and further trained on the remaining number of epochs. For example, the larger models (AraRoBERTa-SA, AraRoBERTa-EG) are trained on three epochs. Then the models are saved, and the training process continues on these models on four epochs. Followed by a final three epochs making the total of epochs ten. The medium-sized models are trained on five epochs followed by another five epochs. In contrast, the smaller models are trained on continuous seven epochs followed by another three epochs or directly ten epochs.

5.3.4 Training Optimization

The optimization is similar to the adopted BERT optimization [118]. An Adam optimizer [145] is adopted using the following parameters shown in Table 5.3. The Beta 1 = 0.9, Beta 2 = 0.999, epsilon = $1e - 08$ and the learning rate is warmed up to a peak value of $5e - 5$.

Hyperparameter	Value
Beta 1	0.9
Beta 2	0.999
Epsilon	$1e - 08$
Learning Rate	$5e - 05$

Table 5.3: AraRoBERTa hyperparameters for Adam optimization algorithm.

5.3.5 Results

It can be noticed in the training loss Figures 5.4-5.10 that with more iterations, the training loss converges with a minor spike when continuing the training of the model trained on seven epochs. This could be due to the random seeds of the model. AraRoBERTa yields a satisfactory performance and, in some cases, better performance compared to other models in the following chapter.

The pre-training process of AraRoBERTa-ALL is illustrated in Figure 5.3. The model is trained over 340,000 steps over 51 hours approximately. AraRoBERTa-ALL is trained on two epochs, and then the pre-trained model is loaded and trained on another two epochs consecutively. AraRoBERTa-SA follows a similar approach as Figure 5.4 illustrates. The model has trained over 161,500 steps for 25 hours, approximately three, four, and a final three epochs, respectively, for ten epochs. Additionally, Figure 5.5 illustrates the pre-training process of AraRoBERTa-EG; the model is trained over 133,500 steps for 22 hours approximately. AraRoBERTa-EG is trained on three epochs, and the pre-trained model is loaded and trained on four epochs and similarly followed by another three epochs, for a total of ten epochs.

Figure 5.6 illustrates the pre-training process of AraRoBERTa-KU; the model is trained over 31,000 steps for several hours. AraRoBERTa-KU is trained on five epochs; then, the pre-trained model is loaded and trained on five epochs for ten epochs. In Figure 5.8, the pretraining process of AraRoBERTa-JO; the model is trained over 9,000 steps for several hours. AraRoBERTa-JO is trained on five epochs; then, the pre-trained model is loaded and trained on five epochs for ten epochs. In

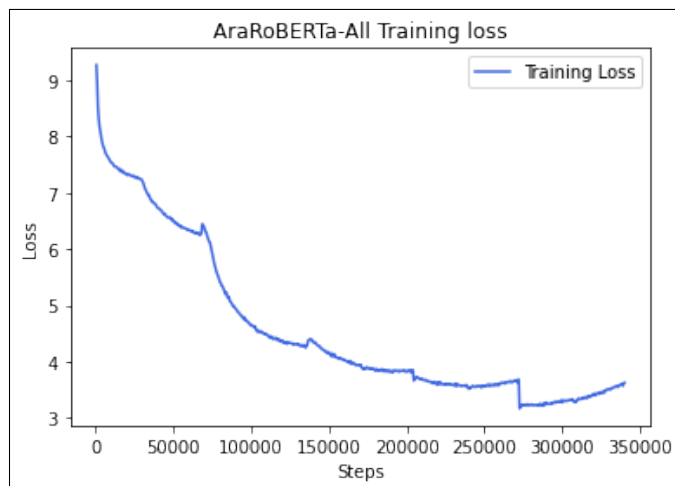


Figure 5.3: Training loss of AraRoBERTa-ALL: 340,000 steps equals 51 hours of split training on 2,2,2,2,2 epochs.

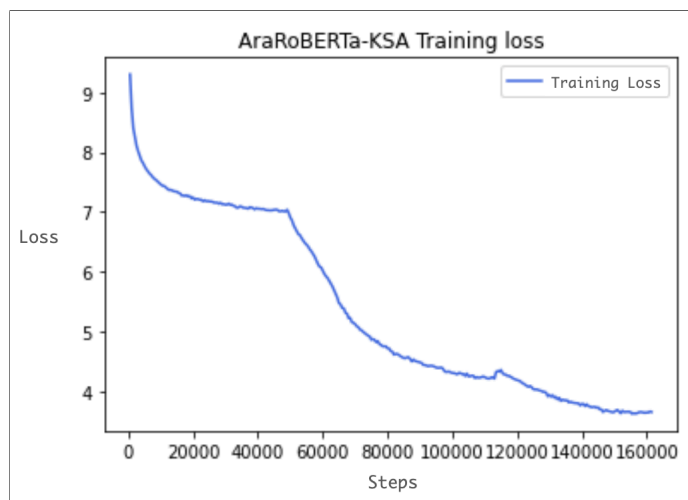


Figure 5.4: Training loss of AraRoBERTa-SA: 161,500 steps equals 25 hours of split training on 3,4,3 epochs.

addition, Figure 5.9 illustrates the pretraining process of AraRoBERTa-LB; the model is trained over 16,000 steps for several hours. AraRoBERTa-LB is trained on ten epochs. Finally, Figure 5.10 illustrates the pretraining process of AraRoBERTa-DZ; the model is trained over 31,000 steps. AraRoBERTa-DZ is trained on seven epochs; next, the pre-trained model is loaded and trained on three epochs for ten epochs.

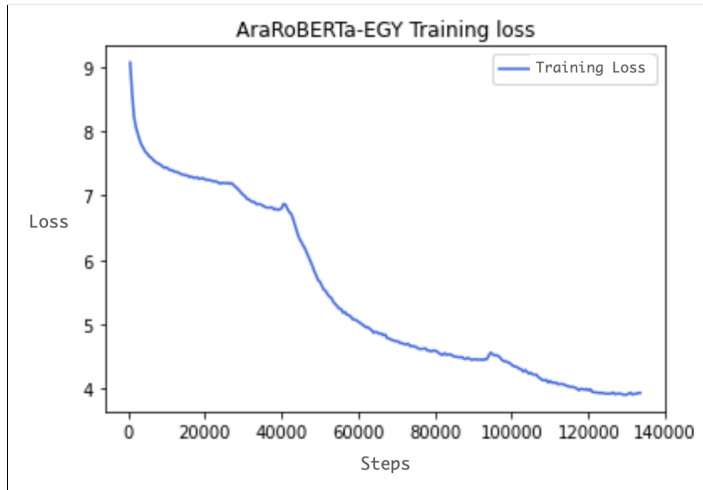


Figure 5.5: Training loss of AraRoBERTa-EG: 133,500 steps equals 22 hours of split training on 3,4,3 epochs.

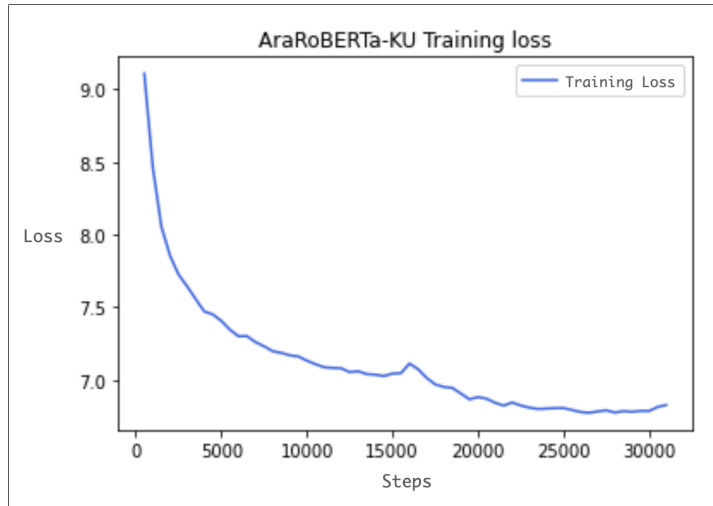


Figure 5.6: Training loss of AraRoBERTa-KU: 31,000 steps equals 6 hours of split training on 5,5 epochs.

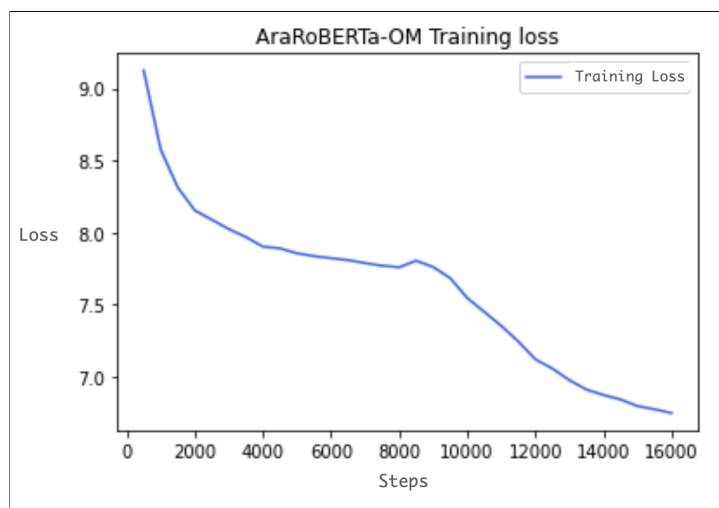


Figure 5.7: Training loss of AraRoBERTa-OM: 16,000 steps equals 7 hours of split training on 5,5 epochs.

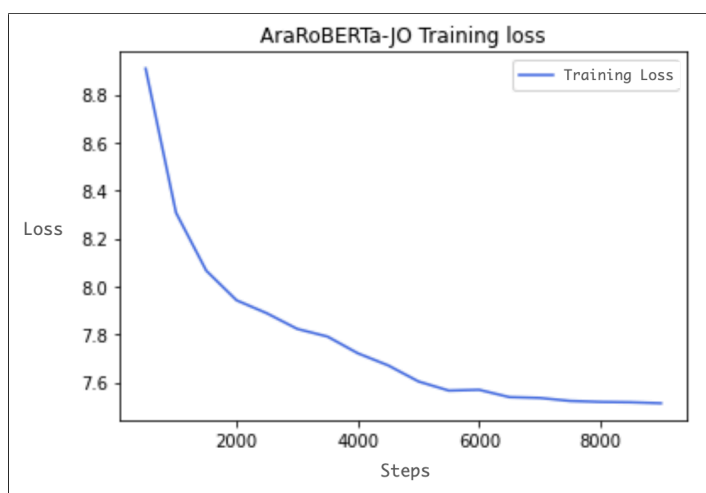


Figure 5.8: Training loss of AraRoBERTa-JO: 9,000 steps equals 4 hours of continuous training on 5,5 epochs.

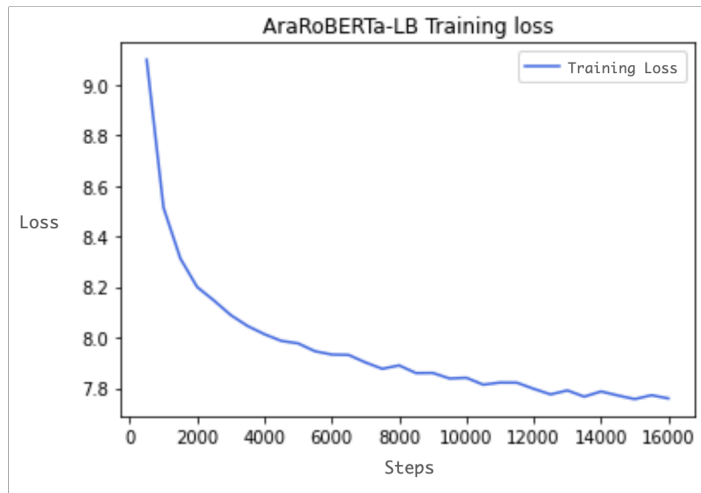


Figure 5.9: Training loss of AraRoBERTa-LB: 16,000 steps equals 3 hours of continuous training on 10 epochs.

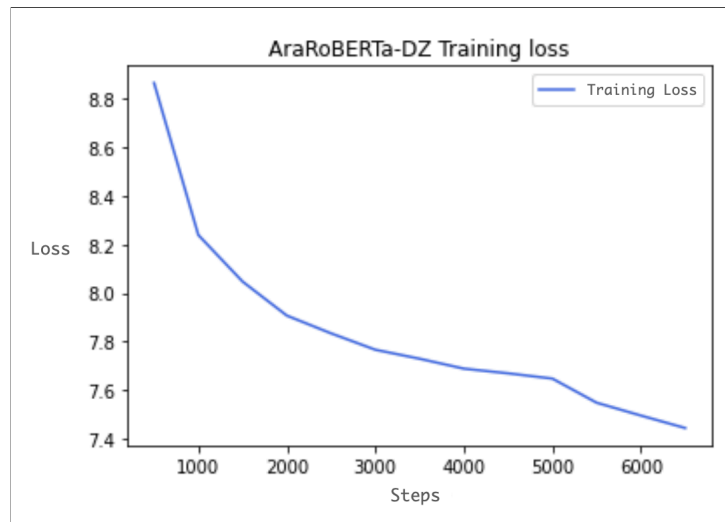


Figure 5.10: Training loss of AraRoBERTa-DZ: 6,500 steps equals 2 hours of continuous training on 7,3 epochs.

CHAPTER 6

ARABIC DIALECT IDENTIFICATION

This chapter covers the proposed approaches for the ADI task and evaluates them.

6.1 Supervised Learning Approach

The learning model in the text classification task is given a text with a predefined class to learn from and predict the class on unseen text [28]. The supervised ADI in text is the process of identifying the dialect of a sentence or a document [24]. In other words, it is the process of developing a classifier that can recognize a specific text T as belonging to a particular dialect D based on the provided annotation tags attached to the text [24], [25]. Although Arabic dialects have the same alphabet and have some shared vocabulary, the dialects have unique characteristics that can aid in the dialect classification task [24]. These characteristics are considered as challenges since they require developing dialect-specific resources. However, it is what makes these dialects

distinguishable.

Supervised learning requires annotated examples to learn from. Usually, these examples require human annotation, which requires allocation of resources [146]. PLMs enable learning from data with fewer annotated examples since learning happens in the pre-training phase as well [67]. PLMs can be used in text classification by using a task-specific linear classifier and fine-tuning it on task-specific labels, in our case, Arabic dialect labels [27]. Arabic suffers from the lack of annotated datasets, and using PLMs could mitigate the lack of domain-specific labels and leverage data from similar domains, namely, Arabic dialect text.

6.1.1 Classification Model

The process of developing a supervised dialect classifier, as Figure 6.1 illustrates, is straightforward. The AraRoBERTa takes as an input the manually labeled data (D_l) that is considered the ground truth text described in Section 4.6.1.

6.1.2 Evaluation

In supervised ADI, the fine-tuning process is an end-to-end process. This work follows a setup similar to the pre-training task described in Section 5.3.3 except for the number of epochs. The number of epochs tends to be larger during the pre-training phase compared to the fine-tuning task [66]. In BERT, the number of fine-tuning epochs is from three to six epochs, and after initial experimentation, the adopted number of epochs is different. The following sections detail the experimental design.

In this experiment, the performance of AraRoBERTa variations on the classifica-

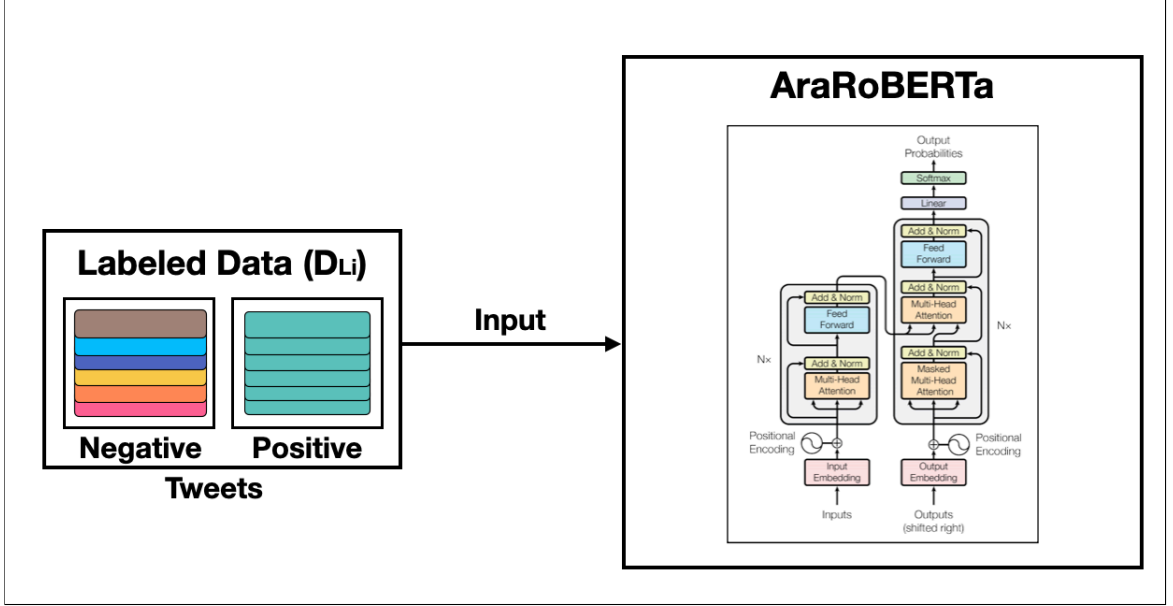


Figure 6.1: Supervised ADI on hand-labeled data.

tion task is evaluated using the manually annotated data described in Section 4.6.1. Then the results are compared to supervised machine learning models. Five epochs are adopted in this setting after initial experimentation. The following sections describe the utilized data and other baselines.

1. Dataset

The dataset used in the supervised text classification task is described in Section 4.6.1. The train/validation/test split is 70/10/20% respectively.

2. Supervised Machine Learning Baseline Models

In order to evaluate the performance of our baseline model on ADI, we utilize traditional Machine Learning (ML) models to compare its performance against our approach. The used models are Logistic Regression (LR) and Support Vector Machine (SVM) since in previous work in the ADI task, they yielded the best results [147].

The models are implemented using *sklearn* Python package with linear kernels and L2 regularization to avoid overfitting. The training data is similar to the described above. The Term Frequency-Inverse Document Frequency (TF-IDF) is used to represent the text. The experiment is performed with a 10-fold cross-validation and a train/test ratio of 80/20.

Results

The evaluation of AraRoBERTa models on the supervised text classification is compared against traditional supervised classification models. In Table 6.1 The results show that the larger AraRoBERTa models, AraRoBERTa-SA and AraRoBERTa-EG, outperform traditional models. Also, AraRoBERTa-KU yields slightly lower results. However, it is clear from the results that traditional ML performs better with a smaller dataset size.

Dialect	AraRoBERTa*	SVM	LR
SA	0.84	0.82	0.79
EG	0.93	0.91	0.86
KU	0.92	0.93	0.92
DZ	0.86	0.93	0.92
LB	0.85	0.90	0.89
JO	0.85	0.89	0.88
OM	0.72	0.89	0.88

Table 6.1: AraRoBERTa models performance against traditional ML model. * denotes the different AraRoBERTa variations corresponding to each dialect.

6.2 Weakly-supervised Learning Approach

The cost of curating, preprocessing, and annotating large datasets that can be utilized in machine learning and deep learning can grow significantly in term cost [148]. The

shortage of annotated data poses a significant challenge for studies working in Arabic. Additionally, in some cases, the data privacy constraints prevent manual inspection of data [148]. Studies have adopted different methods to address these problems. Some works adopt automatic annotation methods; others adopt varying degrees of supervision. This section focuses on Weak Supervised Learning (WSL) to evaluate the automatic annotation approach described in Section 4.7.2.

In weak-supervision, the models are learning from data that are automatically labeled, where the labeling of data is less accurate [149]. Since recruiting a Subject-matter expert (SME) to label can be expensive, other techniques are utilized to approximate the labels. The *inexact supervision* occurs when some information is given that provides *some* supervision not as desired [148], [150]. Although this type of learning is considered weak, it is better in such scenarios where labeled data are limited.

6.2.1 Classification Models

One of the challenges when adopting weak supervised learning is noise; hence, two WSL setups are investigated in this work. The first setup utilizes the data directly after the annotation phase and performs the ADI task. The second setup works to filter the data further from noisy samples and then perform the ADI task.

ADI Model: WSL-I

This model is similar to the supervised learning model in terms of training. However, the difference is in the training data. The data is manually annotated in supervised learning, providing a fully supervised setup. Meanwhile, in weak supervision, the

difference is in the labeled data, as Figure 6.2 illustrates the process of WSL-I in ADI. The key difference is in the annotation phase, where the data is labeled automatically using a labeling algorithm that utilizes an external data source that is a dialect-specific dictionary to provide an approximate label for the unlabelled text as described in Section 4.7.2.

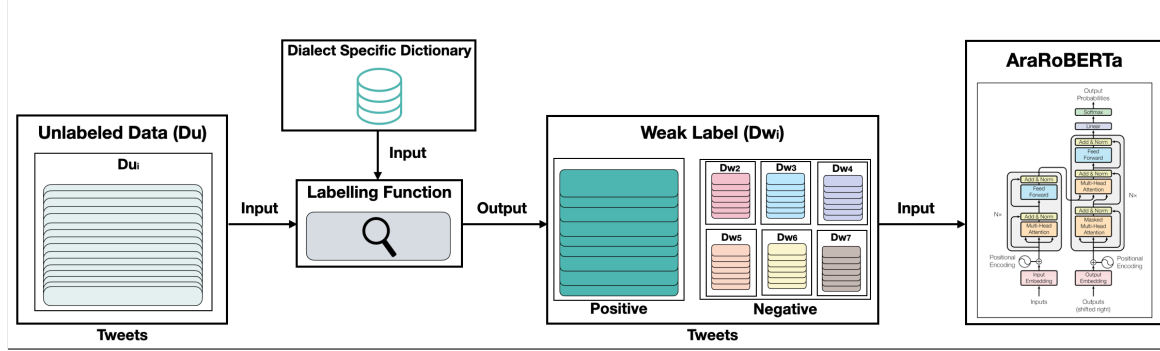


Figure 6.2: The WSL-I model for ADI task.

ADI Model: WSL-II

This model is similar to the previous model; however, after the annotation phase, another phase is proposed to reduce noisy samples in the weakly supervised setup. After going through the weakly labeled data manually, it is noticed that the majority of the noise is MSA prayers, as shown in Figure 4.10. A filtration step is introduced to reduce the prayer content, as Figure 6.3 illustrates. In order to reduce prayer content, 7 specialized user accounts that only produce prayer entries are manually selected. The entries in each account are crawled, collecting 18,222 tweets, and after cleaning and preprocessing, it is reduced to 17,087 tweets. Prayers have a sequential nature; hence, a bigram model is built, and the 100 most frequent bigrams are used to capture prayer tweets and remove them to reduce noise. Then the remaining data is utilized

in the training phase.

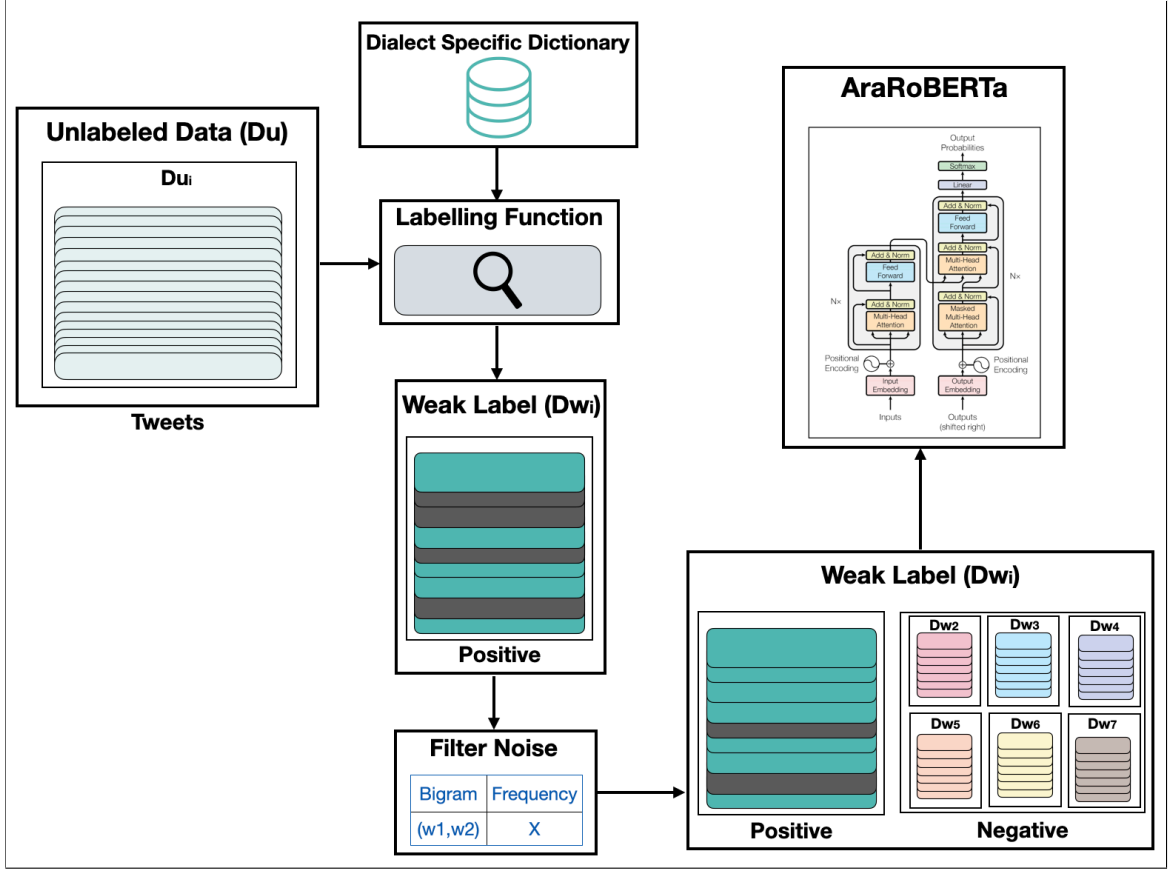


Figure 6.3: The WSL-II model for ADI task.

6.2.2 Evaluation

Weakly supervised dialect classification is similar to supervised dialect classification. It utilizes labeled data to perform the dialect classification task. This task aims to evaluate the effectiveness of this learning approach in the ADI task and compare its performance against the supervised AraRoBERTa classifiers. The key difference between the supervised and weak supervised is in the assigned label, wherein weak-supervision the label is denoted as *weak label* since it is an approximation for the label that is prone to error. The setup for both WSL models is similar; however, the

difference is in the training data as described below.

WSL-I: Dataset

The utilized dataset is described in Section 4.7.2 and the train/validation split is 90/10 respectively as shown in Table 6.2. The test data is the same ground truth data utilized to evaluate the supervised models in Section 6.1.2.

Dialect	Train	Validation
SA	94,489	10,498
KU	47,755	5,306
OM	26,699	29,66
JO	15,746	1,749
LB	18,281	2,031
EG	94,489	10,498
DZ	55,674	6,186

Table 6.2: The training data split for WSL-I.

WSL-II: Dataset

The dataset is similar to WSL-I data; however, in this setup, the amount of data is smaller since the data is filtered from some noisy samples; Table 6.3 shows the training and validation data size.

Dialect	Train	Validation
SA	78,138	8,682
KU	35,534	3,948
OM	18,716	2,078
JO	13,018	1,446
LB	17,478	1,942
EG	78,138	8,682
DZ	40,586	4508

Table 6.3: The training data split for WSL-II.

Setup

The setup in both WSL models is similar to that of the pre-training, and supervised setup described earlier, with a difference in the number of epochs; after initial experimentation, the selected number of epochs is three for both models.

Results

This task aims to evaluate the effectiveness of the developed annotation approaches for labeling the data using AraRoBERTa variations and comparing their performance against the supervised AraRoBERTa classifiers. The weak-supervised dialect classifier results are shown in Table 6.4. Across all models in this setup, it yielded lower performance with varying degrees. In this setup, the training data is larger for all classification models. For example, in the Jordan dialect, the data is around 6x larger, and in the Saudi dialect up to 33x larger. Better performance is noticed in larger AraRoBERTa models, namely, AraRoBERTa-SA and AraRoBERTa-EG, yield better results than other supervised classifiers for other dialects.

The performance of WSL-II increases in four classifiers and decreases in two classifiers. This indicates that adding rules for reducing noise from the dialect samples can effectively enhance the performance. Also, the performance increase indicates that the removed samples contain code-switching between prayers and dialectal content, leading to reduced performance. This approach could lead to a better labeling function that produces representative dialect samples with minimal resource requirements for annotation.

To extract more insights about the factors that play a role in the performance of

LM	Supervised	WSL-I	WSL-II
AraRoBERTa-SA	0.84	0.81	0.82
AraRoBERTa-EG	0.93	0.86	0.87
AraRoBERTa-KU	0.92	0.61	0.60
AraRoBERTa-DZ	0.86	0.78	0.78
AraRoBERTa-LB	0.85	0.78	0.79
AraRoBERTa-JO	0.85	0.71	0.68
AraRoBERTa-OM	0.72	0.40	0.43

Table 6.4: Weak-supervised dialect classification results. Best results are in bold and blue cell. The green cell indicate the best results between the weak classifiers.

the weak-supervised AraRoBERTa models, the models’ performance on the validation set is compared against the performance on the testing set as Table 6.5 shows. The performance on the validation set indicates that the models learned from the weakly labeled data and obtained good results. However, the performance on the test data is lower. To further understand the source of confusion for the models, we look at the predictions of each classifier as Table 6.6 shows. It can be noticed that all models are confused in identifying positive samples except in the SA model.

Dialect	SA	EG	KU	OM	LB	JO	DZ
Validation	0.9	0.96	0.95	0.92	0.97	0.88	0.93
Test	0.81	0.86	0.61	0.40	0.78	0.71	0.78
Performance Change	−10%	−10.4%	−22.1%	−56.5%	−19.6%	−19.3%	−16.1%

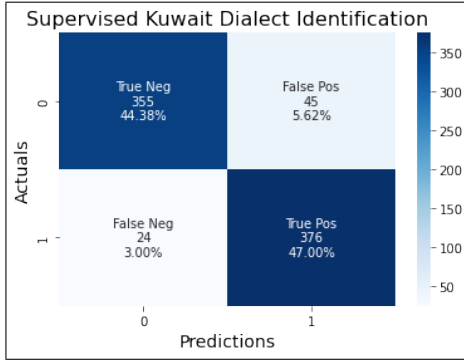
Table 6.5: The WSL-I performance on the validation and test sets.

	True Negative	False Positive	False Negative	True Positive
SA	303	97	60	340
EG	372	28	79	321
DZ	346	54	112	288
LB	391	9	137	263
JO	349	51	153	247
KU	390	10	221	179
OM	377	23	293	107

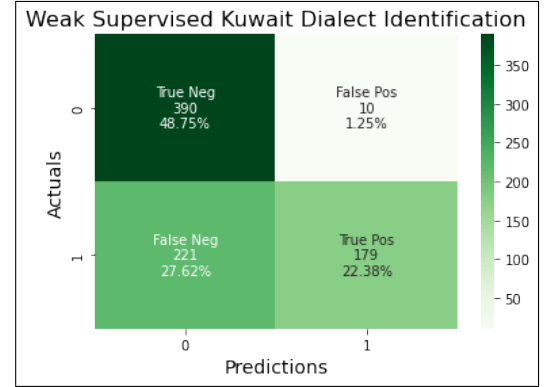
Table 6.6: The predictions of WSL-I AraRoBERTa Classifier. The Highest Incorrect Predictions are bold.

When comparing the supervised AraRoBERTa-KU classifier with the WSL-I

AraRoBERTa-KU classifier as Figure shown in 6.4, the WSL-I model predicts the weak positive sample as a negative sample, indicating that these samples are noisy since the supervised version can identify the positive samples easily, which can be one of the downsides of utilizing this approach. On the other hand, the WSL-I model is effective in different dialects such as SA and EG on the same task. This provides a promising way for automatically labeling the dialect when having a pretrained model trained on large data such as SA and EG.



(a) The misclassification in AraRoBERTa-KU. The true positive is higher than the true negative, indicating the ability of the model to identify KU dialect.



(b) The misclassification in AraRoBERTa-KU. The true negative is high, indicating the model is better in identifying the non-KU dialect.

Figure 6.4: Confusion matrix for AraRoBERTa-KU in the supervised and weak supervised setup.

The noise in the labeled entries motivated the filtration process adopted in the WSL-II approach. The results vary after training AraRoBERTa variations on the new training data, as Table 6.7 shows. It can be noticed that there is a performance increase in four classifiers. Other classifiers reported lower results, indicating that good dialectal samples are removed in the process. This is due to the code-switching problem in Arabic dialects where different dialects are mixed in the same tweet, as

noticed earlier in Figure 7.3b.

Dialect	SA	EG	KU	OM	LB	JO	DZ
WSL-I	0.81	0.86	0.61	0.4	0.78	0.71	0.78
WSL-II	0.82	0.87	0.6	0.43	0.78	0.71	0.78
Performance Change	+1%	+1%	-1%	+3%	0%	0%	0%

Table 6.7: The performance change between WSL-I and WSL-II models.

6.3 Semi-supervised Learning Approach

Working on *supervised learning* tasks requires having access to annotated data where the learning algorithm can map point x to its corresponding label y [151]. In order to obtain annotated data, the annotation phase requires recruiting domain experts, which can be costly from the resources point of view. On the other hand, *unsupervised learning* does not require annotated data. It works on learning the underlying patterns from the input data [151]. For example, clustering works on mapping data points to groups of similar inputs in the same group [151].

Semi-Supervised Learning (SSL) aims to combine both the supervised and unsupervised learning tasks to improve the performance [152], [153]. The SSL algorithm takes the unannotated data coupled with some supervision from labeled instances [152]. Semi-supervised learning is adopted in both the classification and clustering methods [152]. However, a large body of literature focuses on adopting SSL in classification, especially when the labeled data is scarce [151]. In similar cases, it can be challenging to construct a reliable supervised classifier due to the lack of annotated data or difficulty in obtaining it; hence, SSL is utilized in the ADI task [151].

Problem Definition in SSL

The supervised and WSL are similar in the ADI task; the difference is in the label's strength or reliability. However, in SSL, the ADI definition slightly differs as follows: given a set of instances in the input data

$$D = \{(x_1, y_1), \dots (x_l, y_l), (x_{u_{l+1}}), \dots, (x_{u_n})\}$$

where $x_i \in X$, and X denotes the input instances features, and $y_i \in Y$ where $y_i \in Y$, and $Y = \{P, N\}$ denotes the label for the instances. The number of instances is denoted by n , the number of labeled instances is denoted by l , and the number of the unlabeled instances $u = n - l$ where u denotes the unlabeled instances.

The SSL aims to learn from both the labeled and unlabeled data by labeling the unlabeled data points to provide a pseudo-label $y_s \in Y$ where the learning becomes

$$D = \{(x_1, y_1), \dots (x_l, y_l), (x_{s_{l+1}}), \dots, (x_{s_n})\}$$

The learning task becomes $f' : X_{l+s} \Rightarrow Y$, where f' denotes learning from labeled data denoted by X_l and pseudo-labeled data denoted as X_s . The supervised learning task is denoted by $f : X_l \Rightarrow Y$ since the learning task is done only on the ground truth data.

6.3.1 Classification Models

This thesis adopts two semi-supervised approaches where the difference is the data. SSL-I refers to the first approach, and SSL-II refers to the second, which are described below.

ADI Model: SSL-I

The SSL-I is a wrapper method that is an inductive-based semi-supervised learning method. It is one of the earliest methods in semi-supervised learning approaches and is also known as *self-training*, *self-labeling*, or *decision-directed learning* [152]. It appeared in the early literature of SSL, where the learning process is iterative. It works by labeling the unlabeled data and incorporating it in the new iteration of learning since [154], [155], [156]. The semi-supervised setting is based on extending the supervised learning algorithms.

First, it trains the classifier on the labeled data. It utilizes this model in generating a *pseudo-label* for the unlabelled data, and the process iterates as Algorithm 3 shows [151]. At each step, according to the current decision function, a subset of the unlabeled data is labeled, then the classifier is retrained on both the labeled and pseudo-labeled data. The final developed inductive classifier does not distinguish between the labeled and the augmented unlabeled data [151], [157], [152].

Algorithm 3: Semi-supervised Algorithm

```
1 Semi-supervised ( $D_L, D_U$ );  
   Input : Labeled data  $D_L$ , Unlabeled data  $D_U$   
   Output: Semi-labeled data  $D_S$   
2  $D_L = (\mathbf{t}_i, y_i)_{i=1}^l$   
3  $D_U = (\mathbf{t}_j)_{j=1}^u$   
4  $f = \text{train.fit}(D_L)$   
5 for  $t_j \in D_U$  do  
6    $t_j.\text{label} = f.\text{predict}(t_j);$   
7   if  $t_j.\text{label.confidence} > \text{threshold}$  then  
8      $D_S \leftarrow t_j;$   
9      $D_U$  remove  $t_j;$   
10 end  
11 return  $D_S$ 
```

Figure 6.5 illustrates our proposed approach in this work. The classifier algorithm is built by using the developed language model. The classifier is trained to learn from the labeled data, and then the trained model is used to label the unlabeled data. The unlabeled data is sampled 60% from a specific dialect that is collected from a specific country D_{ui} corresponding to the investigated dialect since it contains noise as shown in Section 4.3.4; 40% of the unlabeled sample is incorporated from other unlabeled dialects collected from other countries. Producing a pseudo-label that is fed again for the model to train if the confidence level of the label is greater than 95% both the negative and positive examples are combined with the previously labeled data and considered as the new training examples that are unbalanced. The process stops when the remaining unlabeled data is less than 5% of the original unlabeled data size, or no remaining data is left to label.

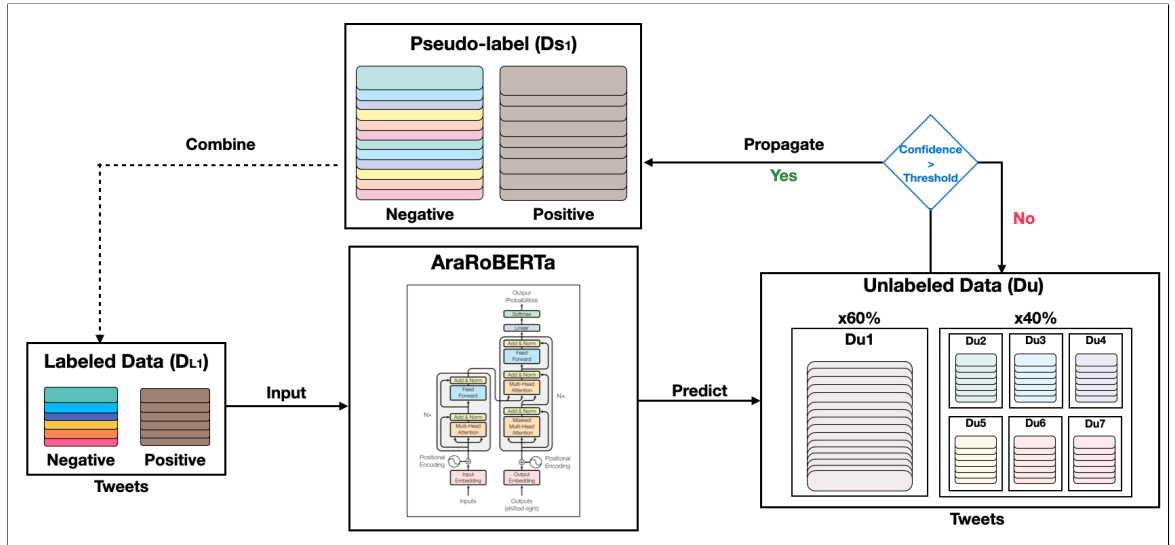


Figure 6.5: SSL-I model for ADI task.

ADI Model: SSL-II

The SSL-II is a variation of SSL-I to produce balanced training data. The SSL-II incorporates the positive pseudo-labeled data from other dialects as negative samples. The process in SSL-II takes the positive pseudo-labeled data with high confidence from other classifiers and incorporates it as a negative class to learn from. In the previous approach, the negative sample is incorporated from the pseudo-labeled data produced by the classifier based on the sample of unlabeled data that is sampled from different unlabeled data. However, the produced pseudo-label training data is unbalanced. Also, the produced negative sample is prone to error since the data is unlabeled and might contain dialectal content that belongs to the positive sample. Therefore we proposed the SSL-II model to mitigate this limitation as Figure 6.6 illustrates.

6.3.2 Evaluation

The semi-supervised setup follows an approach similar to the weak-supervised setup in Section 6.2.2. In this experiment, a combination of labeled and unlabeled entries previously described in Section 4.6.1 and Section 4.5.1 respectively are used. The sample size is reduced due to computational limitations; a random sample of 90,000 training samples is selected to perform the semi-supervised experiments with a confidence of 0.9 for classification.

SSL-I

This setup is similar to the previous setup. However, the initial size of the utilized data is reduced to 16,000 training samples that are randomly selected. The random

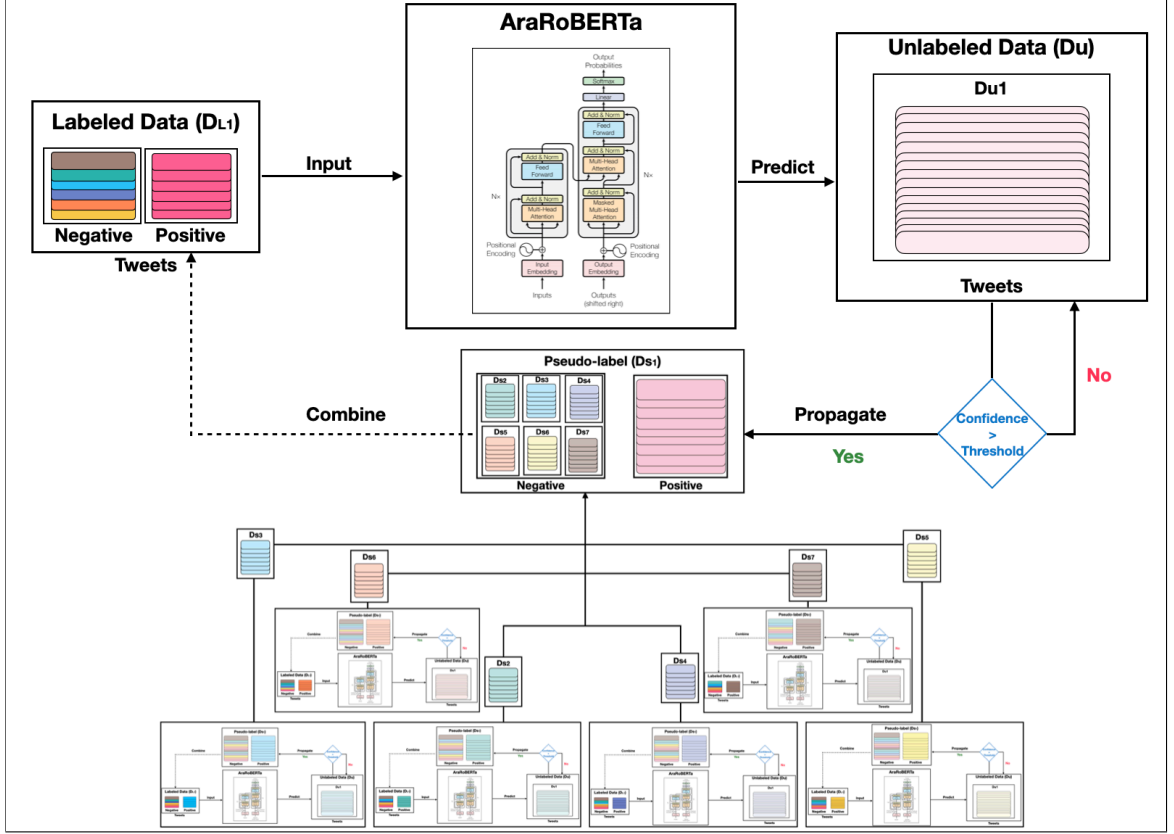


Figure 6.6: The SSL-II model for ADI task.

sampling is 60/40, where 60% of the sample is from the specifically investigated dialect, and the rest is a uniform sample from other dialects. The threshold for the prediction confidence is increased to 0.95.

SSL-II

Here it is similar to SSL-I, with a slight difference in the negative sample where it is incorporated from other groups of dialects. Each model predicts the positive samples incorporated into another model as a negative sample. The negative sample at each iteration is calculated by following this formula:

$$Semi_{Neg} = \sum_{n=1}^{n=6} \frac{Semi-Pos.n}{6}$$

The $Semi_{Neg}$ denotes the negative samples from the other dialects and $Semi -$

Positive represents the value of the negative samples that should be taken from each dialect in order to produce a balanced training set. The sum of these small portions assembles a size equal to the semi-positive samples, creating a balanced training dataset. Table 6.8 shows the negative sample at each iteration in the proposed approach. The *semi-negative* sample equals $(semi-positive/6)$, the *semi-balanced* equals $(semi-positive + semi-negative)$ and the new training data equals $(semi-balanced + training data in iteration_{n-1})$.

LM	Iteration	Semi-Positive	Semi-Negative							Semi-balanced	New Data
			SA	EG	KU	OM	JO	LB	DZ		
AraRoBERTa-SA	1	12,864	-	2,144	2,144	2,144	2,144	2,144	2,144	25,728	28,528
	2	750	-	125	125	125	125	125	125	1,500	1,500
AraRoBERTa-EG	1	7,110	1,185	-	1,185	1,185	1,185	1,185	1,185	14,220	17,020
	2	3,294	549	-	549	549	549	549	549	6,588	23,608
AraRoBERTa-KU	1	7,308	1,218	1,218	-	1,218	1,218	1,218	1,218	14,616	17,416
	2	2,502	417	417	-	417	417	417	417	5,004	22,420
AraRoBERTa-OM	1	7,242	1,207	1,207	1,207	-	1,207	1,207	1,207	14,484	17,284
	2	2,640	440	440	440	-	440	440	440	5,280	22,574
AraRoBERTa-JO	1	8,844	1,474	1,474	1,474	1,474	-	1,474	1,474	17,688	20,488
	2	3,268	544	544	544	544	-	544	544	6,528	27,016
AraRoBERTa-LB	1	10,320	1,720	1,720	1,720	1,720	1,720	-	1,720	20,640	23,440
	2	2,244	374	374	374	374	374	-	374	4,488	27,928
AraRoBERTa-DZ	1	12,108	2,018	2,018	2,018	2,018	2,018	2,018	-	24,216	27,016
	2	1,297	216	216	216	216	216	216	-	2,592	29,608

Table 6.8: The produced data during the SSL-II training.

Results

Experiments are conducted to evaluate the effectiveness of learning from both labeled and unlabeled data in the dialect classification task. The evaluation compares the performance against the supervised AraRoBERTa classifier’s performance. The results in *iteration-0* are considered supervised since the initial iteration is on the labeled data. The following are semi-supervised iterations since the pseudo-labeled samples are incorporated.

The SSL classifiers, as Table 6.9 shows, outperform the supervised classifiers in

three models and yield similar results on the AraRoBERTa-EG model, indicating the effectiveness of SSL on a low-resourced setup on the dialect classification task. The models yield stable performance in both SSL classifiers, indicating this method’s reliability. The SSL-II classifier provides better performance than SSL-I in five dialects. This shows the effectiveness of the proposed approach in enhancing the performance since the data is balanced, allowing the model to learn from positive and negative samples equally.

LM	Supervised	SSL-I	SSL-II
AraRoBERTa-SA	0.84	0.81	0.83
AraRoBERTa-EG	0.93	0.92	0.93
AraRoBERTa-KU	0.92	0.86	0.89
AraRoBERTa-DZ	0.86	0.82	0.87
AraRoBERTa-LB	0.85	0.86	0.88
AraRoBERTa-JO	0.85	0.83	0.83
AraRoBERTa-OM	0.72	0.85	0.80

Table 6.9: Semi-supervised dialect classification results. Best results are in bold and blue cells. Green cells indicate the best results between the SSL classifiers.

CHAPTER 7

ADI EVALUATION

This chapter discusses the contextual representation and the ADI task results.

7.1 Baseline Models

The variation in the training steps between the AraRoBERTa models depends on the training data size and the number of epochs. The number of steps is affected by these variables. The duration of the training is affected by the speed of processing the data from a hardware perspective. The larger PLMs, AraRoBERTa-ALL, AraRoBERTa-SA, and AraRoBERTa-EG, have more significant steps than the other models, increasing the training duration as well. The pretraining can reach 51 hours to develop similar models.

The utilized platform is Google Colab, which provides premium members with the needed computational resources. The allocation of this resource can vary based on availability [158]. In the training logs, the variation between the number of training samples per second where the pretraining from scratch will process fewer instances

during training. However, more samples are processed per step when training a pre-trained model. This can be seen in AraRoBERTa-LB, where training the model from scratch; the number of training samples per second is 15.453, whereas, when training the pretrained model, the corresponding number is 57.088.

In order to evaluate the performance of our baseline model on dialect identification, other existing pretrained models are utilized and fine-tuned on the same dataset with the same setup described in Section 6.1.2. The selected models are either multilingual models trained on multiple languages, including Arabic, or monolingual ones trained only on Arabic. The multilingual models are the original models by the BERT and RoBERTa authors, *mBERT* is the multilingual version of BERT, and *XLM* is the multilingual version of RoBERTa. *AraBERT* is the monolingual model developed only for MSA. The AraRoBERTa-All model is trained on all seven dialects and utilized as a baseline. Table 7.1 provides a comparison between these models based on the training data on Arabic, the vocabulary size, and the model configuration.

Model	Training Data			Vocabulary		Configuration	
	Source	Variant	#Tokens	Tokenizer	Size	Arch.	#Params.
mBERT	Wikipedia	MSA/Multi-Lang	Ar(153M)/All(1.5B)	WP	Ar(5K)/All(110K)	base	110M
XLM- R_B	CommonCrawl	MSA/Multi-Lang	Ar(2.9B)/All(295B)	SP	Ar(14K)/All(250K)	base	270M
AraBERT	Several (3 sources)	MSA	2.5B	SP	Ar(60K)/All(64K)	base	135M
AraRoBERTa-SA	Arabic Twitter	SA DA	45.4M	BPE	52K	base	126M
AraRoBERTa-EG		EG DA	37.2M	BPE	52K	base	126M
AraRoBERTa-KU		KU DA	8.9M	BPE	52K	base	126M
AraRoBERTa-OM		OM DA	3.8M	BPE	52K	base	126M
AraRoBERTa-LB		LB DA	3.6M	BPE	52K	base	126M
AraRoBERTa-JO		JO DA	2.6M	BPE	52K	base	126M
AraRoBERTa-DZ		DZ DA	1.9M	BPE	52K	base	126M
AraRoBERTa-All		DA	58M	BPE	52K	base	126M

Table 7.1: Configurations of existing models and AraRoBERTa models. WP is Word-Piece and SP is SentencePiece

7.2 Experimental Results

The AraRoBERTa variations are compared with the other contextual baselines described earlier. As Table 7.2 shows AraRoBERTa variations developed on the corresponding dialect compared to other contextual models. The AraRoBERTa-SA variety outperforms other contextual baselines except for the AraRoBERTa-All, yielding slightly lower results. Also, AraRoBERTa-EG outperforms other existing models. The AraRoBERTa-KU model outperforms other models and provides similar results to AraRoBERTa-All.

In other cases such as LB and JO, the AraRoBERTa-All yields the best results, outperforming other models, including the AraRoBERTa variations corresponding to each dialect, while in Om and DZ dialects, the XLM model outperforms other models, including AraRoBERTa variations. Although AraRoBERTa models are trained on a maximum of 1.8% of the data that AraBERT is trained on, it yields very competitive results. The remaining two, although trained on a smaller fraction of data, yield a similar performance to AraBERT and multilingual models.

This encourages training other models on a specific content even if the available data size is smaller than other training data in the literature. Also, in five out of seven dialects, AraRoBERTa-All outperformed the contextual baseline models. This motivates the development of a multi-dialectal Arabic model for dialects with lower training data to yield competitive performance on ADI tasks.

Dialect	AraRoBERTa*	AraRoBERTa-All	AraBERT	mBERT	XLM
SA	0.84	0.87	0.81	0.82	0.78
EG	0.93	0.92	0.9	0.87	0.88
KU	0.92	0.92	0.91	0.88	0.89
DZ	0.86	0.84	0.86	0.87	0.90
LB	0.85	0.92	0.85	0.88	0.87
JO	0.85	0.91	0.86	0.87	0.83
OM	0.72	0.87	0.85	0.84	0.90

Table 7.2: The performance of the AraRoBERTa models against other contextual models. * denotes the different AraRoBERTa models corresponding to each dialect.

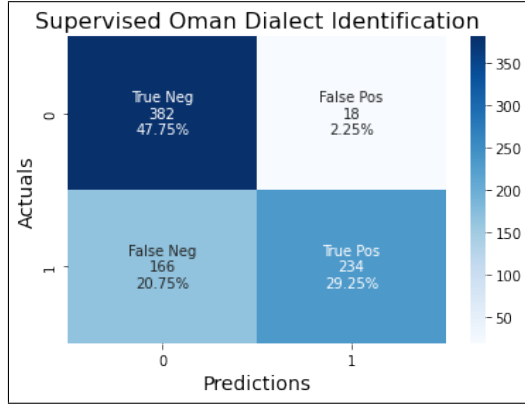
7.3 Discussion

Further evaluation is conducted to better understand the classification model performance by finding the confusion matrix that summarizes predictions of the classifier. This helps in extracting more insights related to the models' performance. The model with the lowest performance in this experiment, AraRoBERTa-OM, Figure 7.1a shows the confusion matrix for AraRoBERTa-OM.

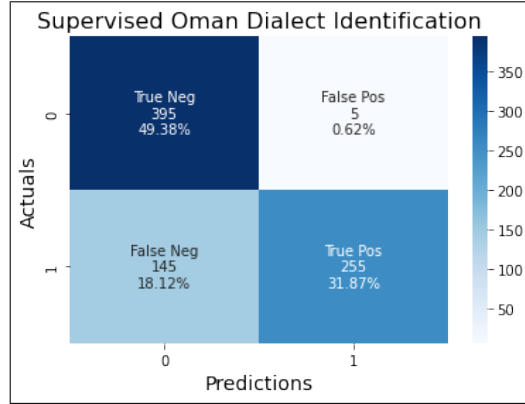
The classifier does not easily identify the Oman dialect from the test samples, although the data is balanced for positive and negative samples. Several positive samples in the test data are identified that might confuse the model. These samples contain vocabulary shared with another dialect or suffer from code-switching between dialects. For example, a tweet containing Oman dialect and MSA led to confusion, making it hard to identify the dialect.

These samples are replaced with other samples that are more representative ; the amount of replaced samples is around 10% of the test data. After evaluating the model on this data, the model is better in identifying the Oman dialect. The previous F-1 score is 0.72, and the current score is 0.77. This is reflected on the confusion matrix

in Figure 7.1b as well. The model’s ability to identify the Oman dialect increased by around 3%, and the true negative increased as well, increasing the correctly classified samples by 34 data points compared to previous results.



(a) Most of the misclassification in AraRoBERTa-OM is within the false negative predictions, indicating the model’s confusion when identifying the Oman dialect.



(b) Removing noisy samples increased the model performance; both the true positive and negative increased and false positive and negative decreased.

Figure 7.1: Confusion matrix for AraRoBERTa-OM.

Other models such as AraRoBERTa-SA and AraRoBERTa-LB are confused in identifying the negative sample as shown in Figures 7.2 and 7.3. In both cases, the reason behind the misclassification could be the fluidity between dialects, as Figure 7.2b shows the misclassified samples. Although the tweet belongs to another dialect (Jordan dialect), all of the tweets’ words except one can be used by Saudi speakers. Figure 7.3b shows a similar situation.

In addition, the first data sample combines two dialects since a speaker from the Jordanian dialect speaks about a song in the Egyptian dialect. Also, AraRoBERTa-LB confuses the Jordan dialect with the Lebanese dialect since both belong to the same geographical region, affecting these dialects’ closeness. Although the Jordan dialect is

from a different region than the Saudi dialect, the effect of geographical proximity on the dialect can be seen where both share geographical borders.

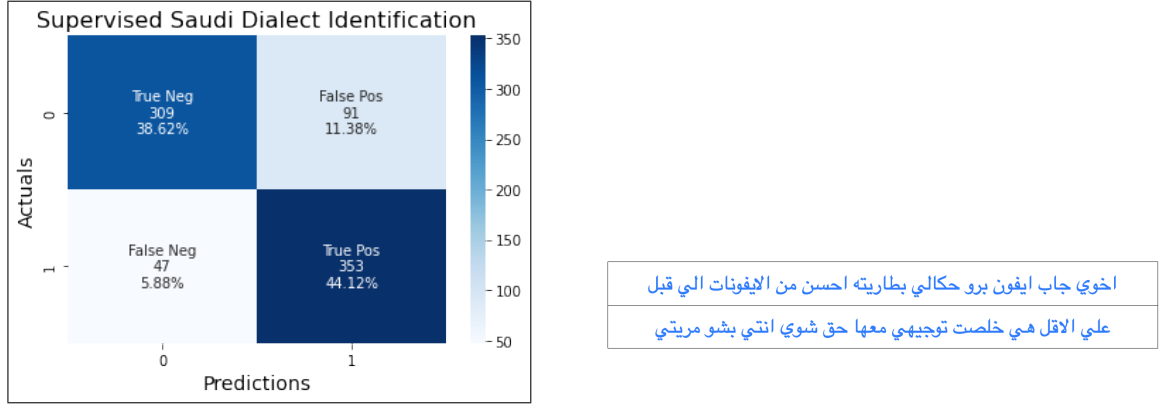


Figure 7.2: AraRoBERTa-SA confusion matrix and a misclassified sample.

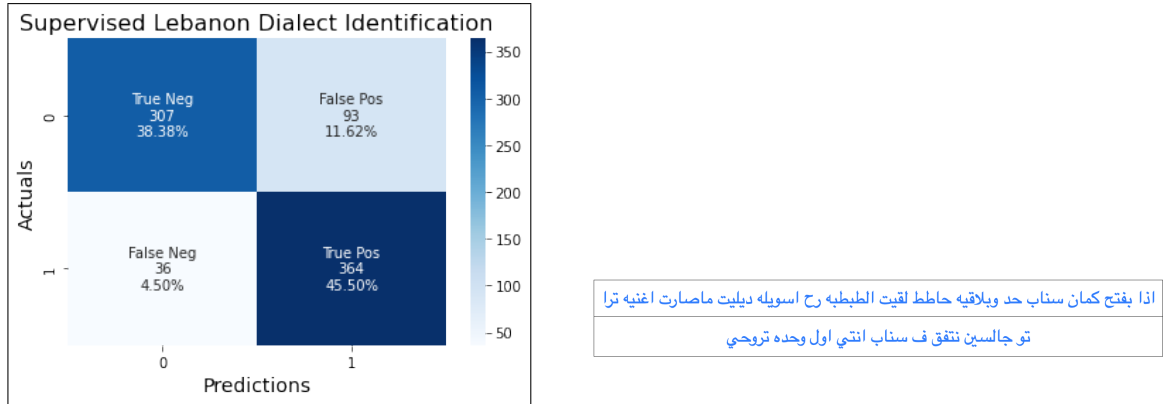


Figure 7.3: AraRoBERTa-LB confusion matrix and a misclassified sample.

Other models yielded good results where the performance of these models to identify both the true negative and true positive is high. The figures below show the ability to identify the dialect with more than 40% accuracy on true negative and true positive samples. Figure 7.6 shows AraRoBERTa-EG yielding the highest results with

more than 46% accuracy in identifying the true negative and true positive samples. It can be inferred that the Egyptian dialect is the least similar to other dialects, reducing the model confusion and increasing the model’s ability to learn.

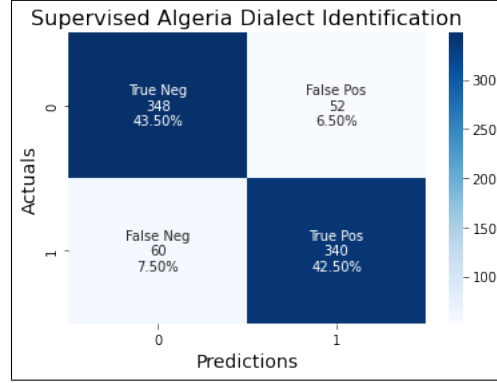


Figure 7.4: AraRoBERTa-DZ confusion matrix, the misclassification in false positive and negative predictions, and the true positive and negative are comparable.

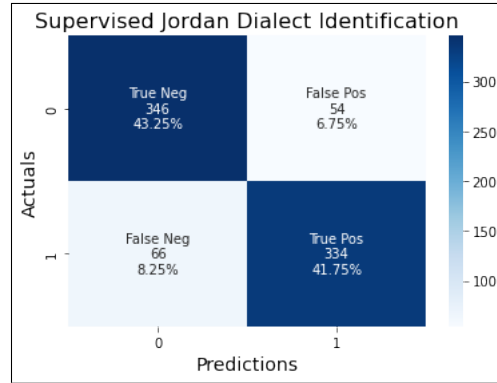


Figure 7.5: AraRoBERTa-JO confusion matrix, the misclassification is in false positive and negative, and true positive and negative sample predictions are comparable.

The fluidity among dialects is a challenge when working on the dialect identification task, as seen in previous examples. The similarity among dialects, as Figure 7.7 shows, varies from dialect to dialect. The dialects that have closer geographical proximity are similar to each other compared to countries from different geographical regions. Notably, the DZ dialect shows more similarity to the Gulf region dialect. This could be related to the DZ data points since it contains more MSA samples, and the Gulf

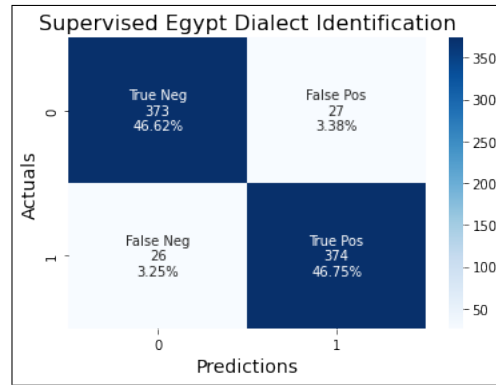


Figure 7.6: AraRoBERTa-EG confusion matrix, the misclassification is in false positive and negative, and true positive and negative sample predictions are comparable.

dialects are the most similar varieties to MSA.

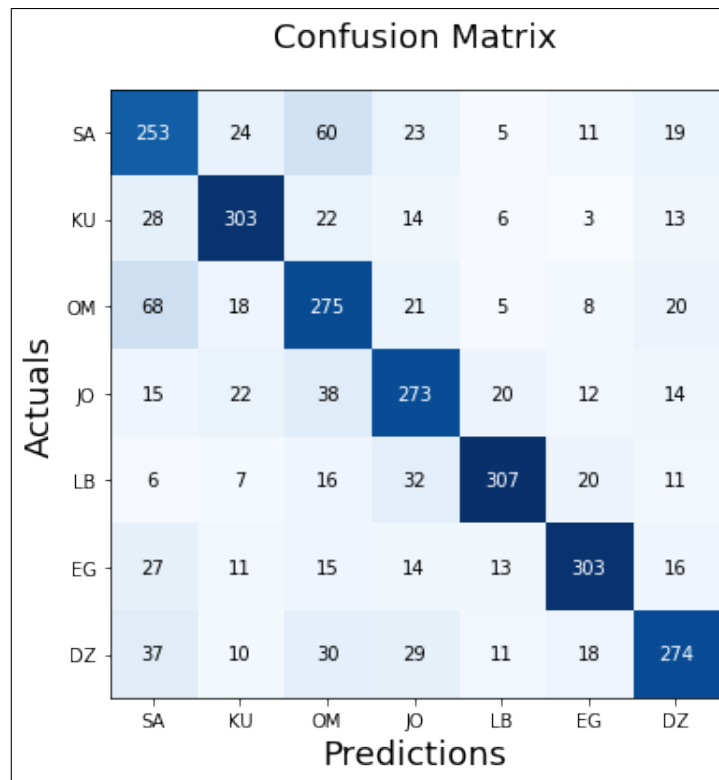


Figure 7.7: Multi-dialect identification confusion matrix.

CHAPTER 8

CONCLUSION AND FUTURE WORK

This chapter covers the conclusion of the thesis work and provides future directions on Arabic dialects.

8.1 Conclusion

Developing dialectal contextual language models yields effective performance across dialects as shown in Section 7.2. The performance of AraRoBERTa models compared with the supervised ML models shows that larger AraRoBERTa models outperform the supervised ML-based models. Also, AraRoBERTa models are pretrained on a fraction of the data size of the baseline contextual language models evaluated in this work. However, the results show that most supervised AraRoBERTa models outperform the other baseline models.

Regarding the different adopted approaches for learning from Arabic dialects, the

supervised models showed effectiveness with varying degrees depending on the size of the AraRoBERTa models. In larger AraRoBERTa models, namely, AraRoBERTa-SA and AraRoBERTa-EG, the manually annotated data are sufficient. Also, no noticeable improvement is acquired by increasing the annotated data by adopting automatic annotation in a weak-supervised setup. On the other hand, a noticeable improvement is observed in smaller models when increasing the size of the annotated data in the semi-supervised approaches. The semi-supervised approach is more effective than the weak-supervised approach when learning from dialectal data. Although the weak-supervised setup utilized more training data, the predicted pseudo-label in the semi-supervised settings did have a higher quality, which was reflected in the performance.

8.2 Future Work

The work investigated in this thesis can be extended in the following directions.

8.2.1 Arabic Dialect Resources

The shortage of Arabic dialect resources remains a challenge facing the investigation of Arabic and its dialects. There is a need to collect more data for all dialects to investigate them and develop technologies that can ultimately serve various user demographics.

8.2.2 Arabic Dialect Annotations

There is a pressing need to build more annotated corpora either by conducting an annotation campaign or developing an effective annotation technique for Arabic and its dialect. This work adopted an automatic annotation method for Arabic dialects to address the shortage in annotated Arabic corpora. It aims to eliminate many limitations facing the work focusing on this problem. The objective is to expedite the process of developing more accurate systems with a minimized effort.

8.2.3 Arabic Dialect Similarities

Although this work focuses on identifying Arabic dialects, investigating the similarities between dialects can help better identify them. For example, some similarities between Jordan and Saudi dialects are noticed. Although they share geographical borders, they belong to different regions and are usually not grouped together.

REFERENCES

- [1] “Woeid lookup | find yahoo weather code,” <https://www.findmecity.com/index.html>, (Accessed on 12/01/2022).
- [2] “Google ai blog: Transformer: A novel neural network architecture for language understanding,” <https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html>, (Accessed on 04/11/2021).
- [3] “Radius around a point on a map,” <https://www.freemaptools.com/radius-around-point.htm>, (Accessed on 10/22/2021).
- [4] O. F. Zaidan and C. Callison-Burch, “Arabic dialect identification,” *Computational Linguistics*, vol. 40, no. 1, pp. 171–202, 2014.
- [5] A. Elnagar, S. M. Yagi, A. B. Nassif, I. Shahin, and S. A. Salloum, “Systematic literature review of dialectal arabic: identification and detection,” *IEEE Access*, vol. 9, pp. 31 010–31 042, 2021.
- [6] N. Habash, *Introduction to Arabic Natural Language Processing*, ser. Synthesis digital library of engineering and computer science. Morgan & Claypool

- Publishers, 2010. [Online]. Available: <https://books.google.com.sa/books?id=kRIHCnC74BoC>
- [7] D. Radcliffe and P. Bruni, “State of social media, middle east: 2018,” 2019.
- [8] F. Al-Shargi, A. Kaplan, R. Eskander, N. Habash, and O. Rambow, “Morphologically annotated corpora and morphological analyzers for moroccan and sanaani yemeni arabic,” in *10th Language Resources and Evaluation Conference (LREC 2016)*, 2016.
- [9] R. AlYami and R. AlZaidy, “Arabic dialect identification in social media,” in *2020 3rd International Conference on Computer Applications Information Security (ICCAIS)*, 2020, pp. 1–2.
- [10] M. Abdul-Mageed, C. Zhang, A. Elmadany, and L. Ungar, “Toward micro-dialect identification in diaglossic and code-switched environments,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 5855–5876. [Online]. Available: <https://aclanthology.org/2020.emnlp-main.472>
- [11] M. Zampieri, L. Tan, N. Ljubešić, and J. Tiedemann, “A report on the DSL shared task 2014,” in *Proceedings of the First Workshop on Applying NLP Tools to Similar Languages, Varieties and Dialects*. Dublin, Ireland: Association for Computational Linguistics and Dublin City University, Aug. 2014, pp. 58–67. [Online]. Available: <https://aclanthology.org/W14-5307>

- [12] A. Elnagar and O. Einea, “Brad 1.0: Book reviews in arabic dataset,” in *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*. IEEE, 2016, pp. 1–8.
- [13] H. Bouamor, N. Habash, M. Salameh, W. Zaghouani, O. Rambow, D. Abdulrahim, O. Obeid, S. Khalifa, F. Eryani, A. Erdmann *et al.*, “The madar arabic dialect corpus and lexicon,” in *Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)*, 2018.
- [14] R. Baly, A. Khaddaj, H. Hajj, W. El-Hajj, and K. B. Shaban, “Arsentd-lev: A multi-topic corpus for target-based sentiment analysis in arabic levantine tweets,” *arXiv preprint arXiv:1906.01830*, 2019.
- [15] O. Einea, A. Elnagar, and R. Al Debsi, “Sanad: Single-label arabic news articles dataset for automatic text categorization,” *Data in brief*, vol. 25, p. 104076, 2019.
- [16] A. Elnagar, Y. S. Khalifa, and A. Einea, “Hotel arabic-reviews dataset construction for sentiment analysis applications,” in *Intelligent natural language processing: Trends and applications*. Springer, 2018, pp. 35–52.
- [17] J. Watson, *The Phonology and Morphology of Arabic*, ser. Oxford linguistics. Oxford University Press, 2002. [Online]. Available: <https://books.google.com.sa/books?id=hQsWH9EdPvcC>
- [18] H. Bouamor, S. Hassan, and N. Habash, “The MADAR shared task on Arabic fine-grained dialect identification,” in *Proceedings of the Fourth*

- Arabic Natural Language Processing Workshop*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 199–207. [Online]. Available: <https://www.aclweb.org/anthology/W19-4622>
- [19] C. A. Ferguson, “Diglossia,” *word*, vol. 15, no. 2, pp. 325–340, 1959.
- [20] M. J. Althobaiti, “Automatic arabic dialect identification systems for written texts: A survey,” 2020.
- [21] K. Versteegh, *Arabic Language*. Edinburgh University Press, 2014. [Online]. Available: <https://books.google.com.sa/books?id=RiarBgAAQBAJ>
- [22] J. Aoun, E. Benmamoun, and D. Sportiche, “Agreement, word order, and conjunction in some varieties of arabic,” *Linguistic inquiry*, pp. 195–220, 1994.
- [23] M. Diab and N. Habash, “Arabic dialect processing tutorial,” in *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Tutorial Abstracts*, ser. NAACL-Tutorials ’07. USA: Association for Computational Linguistics, 2007, p. 5–6.
- [24] M. Zampieri and P. Nakov, *Similar Languages, Varieties, and Dialects: A Computational Perspective*, ser. Studies in Natural Language Processing. Cambridge University Press, 2021. [Online]. Available: <https://books.google.com.sa/books?id=kOk9EAAAQBAJ>
- [25] M. J. Althobaiti, “Automatic arabic dialect identification systems for written texts: A survey,” 2020.

- [26] J. Nash, “The survey of pidgin and creole languages,” 2014.
- [27] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, “Deep learning based text classification: A comprehensive review,” *CoRR*, vol. abs/2004.03705, 2020. [Online]. Available: <https://arxiv.org/abs/2004.03705>
- [28] C. Silva and B. Ribeiro, *Inductive Inference for Large Scale Text Classification: Kernel Approaches and Techniques*, ser. Studies in Computational Intelligence. Springer Berlin Heidelberg, 2009. [Online]. Available: <https://books.google.com.sa/books?id=kR624jbojpsC>
- [29] Z. Liu, Y. Lin, and M. Sun, *Representation Learning for Natural Language Processing*. Springer Nature, 2020.
- [30] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, “SuperGLUE: A stickier benchmark for general-purpose language understanding systems,” *arXiv preprint 1905.00537*, 2019.
- [31] P. Rajpurkar, R. Jia, and P. Liang, “Know what you don’t know: Unanswerable questions for squad,” 2018.
- [32] “Building a large annotated corpus of english: the penn treebank,” <https://catalog.ldc.upenn.edu/docs/LDC95T7/cl93.html>, (Accessed on 12/25/2021).
- [33] M. Abdul-Mageed, C. Zhang, H. Bouamor, and N. Habash, “NADI 2020: The First Nuanced Arabic Dialect Identification Shared Task,” in *Proceedings of the Fifth Arabic Natural Language Processing Workshop (WANLP 2020)*, Barcelona, Spain, 2020.

- [34] M. Diab, M. Al-Badrashiny, M. Aminian, M. Attia, H. Elfardy, N. Habash, A. Hawwari, W. Salloum, P. Dasigi, and R. Eskander, “Tharwa: A large scale dialectal Arabic - Standard Arabic - English lexicon,” in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC’14)*. Reykjavik, Iceland: European Language Resources Association (ELRA), May 2014, pp. 3782–3789. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2014/pdf/1161_Paper.pdf
- [35] “Arabic treebank - weblog - linguistic data consortium,” <https://catalog.ldc.upenn.edu/LDC2016T02>, (Accessed on 12/25/2021).
- [36] A. Fraser, J. Xu, and R. Weischedel, “Trec 2002 cross-lingual retrieval at bbn.” Citeseer.
- [37] H. Sajjad, K. Darwish, and Y. Belinkov, “Translating dialectal Arabic to English,” in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 1–6. [Online]. Available: <https://aclanthology.org/P13-2001>
- [38] K. Meftouh, S. Harrat, S. Jamoussi, M. Abbas, and K. Smaili, “Machine translation experiments on PADIC: A parallel Arabic Dialect corpus,” in *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*, Shanghai, China, Oct. 2015, pp. 26–34. [Online]. Available: <https://aclanthology.org/Y15-1004>

- [39] H. Elfardy and M. Diab, “Token level identification of linguistic code switching,” in *Proceedings of COLING 2012: posters*, 2012, pp. 287–296.
- [40] K. Darwish, “Arabizi detection and conversion to arabic,” *arXiv preprint arXiv:1306.6755*, 2013.
- [41] N. Habash, F. Eryani, S. Khalifa, O. Rambow, D. Abdulrahim, A. Erdmann, R. Faraj, W. Zaghouani, H. Bouamor, N. Zalmout *et al.*, “Unified guidelines and resources for arabic dialect orthography,” in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.
- [42] “9.xps,” [https://www.idosi.org/wasj/wasj32\(3\)14/9.pdf](https://www.idosi.org/wasj/wasj32(3)14/9.pdf), (Accessed on 12/25/2021).
- [43] A. Abdelali, K. Darwish, N. Durrani, and H. Mubarak, “Farasa: A fast and furious segmenter for arabic,” in *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Demonstrations*, 2016, pp. 11–16.
- [44] A. Pasha, M. Al-Badrashiny, M. T. Diab, A. El Kholy, R. Eskander, N. Habash, M. Pooleery, O. Rambow, and R. Roth, “Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic.” in *Lrec*, vol. 14, no. 2014. Citeseer, 2014, pp. 1094–1101.
- [45] C. Holes, *Colloquial Arabic of the Gulf and saudi arabia*. Routledge & Kegan Paul Books, 1984.

- [46] —, “Bahraini dialects: sectarian differences exemplified through texts,” *Zeitschrift für arabische Linguistik*, no. 13, pp. 27–67, 1984.
- [47] M. W. Cowell, “A reference grammar of syrian arabic (based on the dialect of damascus). arabic series, number seven.” 1964.
- [48] J. Heath, “Moroccan arabic phonology,” *Phonologies of Asia and Africa (including the Caucasus)*, vol. 1, pp. 205–217, 1997.
- [49] S. Novotney, R. Schwartz, and S. Khudanpur, “Unsupervised arabic dialect adaptation with self-training,” in *Twelfth Annual Conference of the International Speech Communication Association*, 2011.
- [50] N. Garera and D. Yarowsky, “Modeling latent biographic attributes in conversational genres,” in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, 2009, pp. 710–718.
- [51] W. Salloum and N. Habash, “Dialectal to standard arabic paraphrasing to improve arabic-english statistical machine translation,” in *Proceedings of the first workshop on algorithms and resources for modelling of dialects and language varieties*, 2011, pp. 10–21.
- [52] Y. Goldberg and G. Hirst, *Neural Network Methods in Natural Language Processing*. Morgan & Claypool Publishers, 2017.
- [53] D. Jurafsky and J. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech*

- Recognition*, ser. Prentice Hall series in artificial intelligence. Pearson Prentice Hall, 2009. [Online]. Available: <https://books.google.com.sa/books?id=fZmj5UNK8AQC>
- [54] “shannon1948.dvi,” <https://people.math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf>, (Accessed on 08/10/2021).
- [55] R. Mitkov, *The Oxford Handbook of Computational Linguistics*, ser. Oxford Handbooks Series. Oxford University Press, 2021. [Online]. Available: https://books.google.com.sa/books?id=5_j8wQEACAAJ
- [56] M. Joos, “Description of language design,” *The Journal of the Acoustical Society of America*, vol. 22, no. 6, pp. 701–707, 1950. [Online]. Available: <https://doi.org/10.1121/1.1906674>
- [57] Z. S. Harris, “Distributional structure,” *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.
- [58] J. Firth, “A synopsis of linguistic theory 1930-1955,” in *Studies in Linguistic Analysis*. Philological Society, Oxford, 1957, reprinted in Palmer, F. (ed. 1968) *Selected Papers of J. R. Firth*, Longman, Harlow.
- [59] “Chapter3.pdf,” <http://stanford.edu/~jlmcc/papers/PDP/Chapter3.pdf>, (Accessed on 08/11/2021).
- [60] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” 2013.

- [61] T. Mikolov, S. W.-t. Yih, and G. Zweig, “Linguistic regularities in continuous space word representations,” in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2013)*. Association for Computational Linguistics, May 2013. [Online]. Available: <https://www.microsoft.com/en-us/research/publication/linguistic-regularities-in-continuous-space-word-representations/>
- [62] “Google code archive - long-term storage for google code project hosting.” <https://code.google.com/archive/p/word2vec/>, (Accessed on 08/14/2021).
- [63] “Embeddings: Obtaining embeddings | machine learning crash course,” <https://developers.google.com/machine-learning/crash-course/embeddings/obtaining-embeddings?hl=en>, (Accessed on 08/14/2021).
- [64] “Word2vec | tensorflow core,” <https://www.tensorflow.org/tutorials/text/word2vec>, (Accessed on 08/14/2021).
- [65] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep contextualized word representations,” 2018.
- [66] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [67] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, “Pre-trained models for natural language processing: A survey,” *Science China Technological*

- Sciences*, vol. 63, no. 10, p. 1872–1897, Sep 2020. [Online]. Available: <http://dx.doi.org/10.1007/s11431-020-1647-3>
- [68] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *arXiv preprint arXiv:1706.03762*, 2017.
- [69] J. Brownlee, *Deep Learning for Natural Language Processing: Develop Deep Learning Models for your Natural Language Problems*. Machine Learning Mastery, 2017. [Online]. Available: https://books.google.com.sa/books?id=__pmoDwAAQBAJ
- [70] D. Rothman, *Transformers for Natural Language Processing: Build Innovative Deep Neural Network Architectures for NLP with Python, PyTorch, TensorFlow, BERT, RoBERTa, and More*. Packt Publishing, 2021. [Online]. Available: <https://books.google.com.sa/books?id=Ua03zgEACAAJ>
- [71] S. Ravichandiran, *Getting Started with Google BERT: Build and train state-of-the-art natural language processing models using BERT*. Packt Publishing, 2021. [Online]. Available: <https://books.google.com.sa/books?id=CvsWEAAAQBAJ>
- [72] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” 2015.
- [73] S. N. Alyami and S. O. Olatunji, “Application of support vector machine for arabic sentiment classification using twitter-based dataset,” *Journal of Information & Knowledge Management*, vol. 19, no. 01, p. 2040018, 2020.

- [74] N. Al-Twairesh, H. Al-Khalifa, A. Al-Salman, and Y. Al-Ohali, “Arasenti-tweet: A corpus for arabic sentiment analysis of saudi tweets,” *Procedia Computer Science*, vol. 117, pp. 63–72, 2017.
- [75] M. Abdul-Mageed, H. Alhuzali, and M. Elaraby, “You tweet what you speak: A city-level dataset of arabic dialects,” in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.
- [76] N. Al-Twairesh, R. Al-Matham, N. Madi, N. Almugren, A.-H. Al-Aljmi, S. Alshalan, R. Alshalan, N. Alrumayyan, S. Al-Manea, S. Bawazeer *et al.*, “Suar: Towards building a corpus for the saudi dialect,” *Procedia computer science*, vol. 142, pp. 72–82, 2018.
- [77] F. Haouari, M. Hasanain, R. Suwaileh, and T. Elsayed, “Arcov-19: The first arabic covid-19 twitter dataset with propagation networks,” *arXiv preprint arXiv:2004.05861*, 2020.
- [78] N. Al-Twairesh, H. Al-Khalifa, and A. Al-Salman, “Arasenti: large-scale twitter-specific arabic sentiment lexicons,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, pp. 697–705.
- [79] A. Assiri, A. Emam, and H. Al-Dossari, “Saudi twitter corpus for sentiment analysis,” *International Journal of Computer and Information Engineering*, vol. 10, no. 2, pp. 272–275, 2016.

- [80] “• twitter: most users by country | statista,” <https://www.statista.com/statistics/242606/number-of-active-twitter-users-in-selected-countries/>, (Accessed on 07/28/2021).
- [81] K. Abidi, M. A. Menacer, and K. Smaili, “Calyou: A comparable spoken algerian corpus harvested from youtube,” in *18th Annual Conference of the International Communication Association (Interspeech)*, 2017.
- [82] M. Itani, C. Roast, and S. Al-Khayatt, “Corpora for sentiment analysis of arabic text in social media,” in *2017 8th international conference on information and communication systems (ICICS)*. IEEE, 2017, pp. 64–69.
- [83] M. Hasanain, R. Suwaileh, T. Elsayed, M. Kutlu, and H. Almerekhi, “Evetar: building a large-scale multi-task test collection over arabic tweets,” *Information Retrieval Journal*, vol. 21, no. 4, pp. 307–336, 2018.
- [84] T. Zerrouki and A. Balla, “Tashkeela: Novel corpus of arabic vocalized texts, data for auto-diacritization systems,” *Data in brief*, vol. 11, p. 147, 2017.
- [85] “Rate limits: Standard v1.1 | docs | twitter developer platform,” <https://developer.twitter.com/en/docs/twitter-api/v1/rate-limits>, (Accessed on 10/16/2021).
- [86] M. Abdul-Mageed, H. Alhuzali, and M. Elaraby, “You tweet what you speak: A city-level dataset of arabic dialects,” in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

- [87] A. Etman and A. L. Beex, “Language and dialect identification: A survey,” in *2015 SAI intelligent systems conference (IntelliSys)*. IEEE, 2015, pp. 220–231.
- [88] O. Obeid, M. Salameh, H. Bouamor, and N. Habash, “ADIDA: Automatic dialect identification for Arabic,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 6–11. [Online]. Available: <https://www.aclweb.org/anthology/N19-4002>
- [89] O. Zaidan and C. Callison-Burch, “The arabic online commentary dataset: an annotated dataset of informal arabic with high dialectal content,” in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011, pp. 37–41.
- [90] H. Elfardy and M. Diab, “Sentence level dialect identification in Arabic,” in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 456–461. [Online]. Available: <https://aclanthology.org/P13-2081>
- [91] P. Mishra and V. Mujadia, “Arabic dialect identification for travel and Twitter text,” in *Proceedings of the Fourth Arabic Natural Language Processing Workshop*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 234–238. [Online]. Available: <https://aclanthology.org/W19-4628>

- [92] O. Obeid, M. Salameh, H. Bouamor, and N. Habash, “Adida: Automatic dialect identification for arabic,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, 2019, pp. 6–11.
- [93] A. Abdelali, H. Mubarak, Y. Samih, S. Hassan, and K. Darwish, “QADI: Arabic dialect identification in the wild,” in *Proceedings of the Sixth Arabic Natural Language Processing Workshop*. Kyiv, Ukraine (Virtual): Association for Computational Linguistics, Apr. 2021, pp. 1–10. [Online]. Available: <https://aclanthology.org/2021.wanlp-1.1>
- [94] A. El Mekki, A. Alami, H. Alami, A. Khoumsi, and I. Berrada, “Weighted combination of BERT and n-GRAM features for nuanced Arabic dialect identification,” in *Proceedings of the Fifth Arabic Natural Language Processing Workshop*. Barcelona, Spain (Online): Association for Computational Linguistics, Dec. 2020, pp. 268–274. [Online]. Available: <https://aclanthology.org/2020.wanlp-1.27>
- [95] K. Gaanoun and I. Benelallam, “Arabic dialect identification: An Arabic-BERT model with data augmentation and ensembling strategy,” in *Proceedings of the Fifth Arabic Natural Language Processing Workshop*. Barcelona, Spain (Online): Association for Computational Linguistics, Dec. 2020, pp. 275–281. [Online]. Available: <https://aclanthology.org/2020.wanlp-1.28>
- [96] B. Talafha, M. Ali, M. E. Za’ter, H. Seelawi, I. Tuffaha, M. Samir, W. Farhan,

- and H. T. Al-Natsheh, “Multi-dialect arabic bert for country-level dialect identification,” 2020.
- [97] G. de Francony, V. Guichard, P. Joshi, H. Afli, and A. Boucekif, “Hierarchical deep learning for Arabic dialect identification,” in *Proceedings of the Fourth Arabic Natural Language Processing Workshop*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 249–253. [Online]. Available: <https://aclanthology.org/W19-4631>
- [98] K. Abu Kwaik and M. Saad, “ArbDialectID at MADAR shared task 1: Language modelling and ensemble learning for fine grained Arabic dialect identification,” in *Proceedings of the Fourth Arabic Natural Language Processing Workshop*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 254–258. [Online]. Available: <https://aclanthology.org/W19-4632>
- [99] K. Meftouh, K. Abidi, S. Harrat, and K. Smaili, “The SMarT classifier for Arabic fine-grained dialect identification,” in *Proceedings of the Fourth Arabic Natural Language Processing Workshop*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 259–263. [Online]. Available: <https://aclanthology.org/W19-4633>
- [100] A. Ragab, H. Seelawi, M. Samir, A. Mattar, H. Al-Bataineh, M. Zaghloul, A. Mustafa, B. Talafha, A. A. Freihat, and H. Al-Natsheh, “Mawdoo3 AI at MADAR shared task: Arabic fine-grained dialect identification with ensemble learning,” in *Proceedings of the Fourth Arabic Natural Language Processing*

- Workshop*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 244–248. [Online]. Available: <https://aclanthology.org/W19-4630>
- [101] Y. Fares, Z. El-Zanaty, K. Abdel-Salam, M. Ezzeldin, A. Mohamed, K. El-Awaad, and M. Torki, “Arabic dialect identification with deep learning and hybrid frequency based features,” in *Proceedings of the Fourth Arabic Natural Language Processing Workshop*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 224–228. [Online]. Available: <https://aclanthology.org/W19-4626>
- [102] Y. Samih, H. Mubarak, A. Abdelali, M. Attia, M. Eldesouki, and K. Darwish, “QC-GO submission for MADAR shared task: Arabic fine-grained dialect identification,” in *Proceedings of the Fourth Arabic Natural Language Processing Workshop*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 290–294. [Online]. Available: <https://aclanthology.org/W19-4639>
- [103] I. A. Farha and W. Magdy, “Mazajak: An online arabic sentiment analyser,” in *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, 2019, pp. 192–198.
- [104] S. Vajjala, B. Majumder, A. Gupta, and H. Surana, *Practical Natural Language Processing: A Comprehensive Guide to Building Real-World NLP Systems*. O’Reilly Media, 2020. [Online]. Available: <https://books.google.com.sa/books?id=hvrrDwAAQBAJ>

- [105] C. Zhang and M. Abdul-Mageed, “No army, no navy: BERT semi-supervised learning of Arabic dialects,” in *Proceedings of the Fourth Arabic Natural Language Processing Workshop*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 279–284. [Online]. Available: <https://aclanthology.org/W19-4637>
- [106] A. Abdelali, H. Mubarak, Y. Samih, S. Hassan, and K. Darwish, “QADI: Arabic dialect identification in the wild,” in *Proceedings of the Sixth Arabic Natural Language Processing Workshop*. Kyiv, Ukraine (Virtual): Association for Computational Linguistics, Apr. 2021, pp. 1–10. [Online]. Available: <https://aclanthology.org/2021.wanlp-1.1>
- [107] W. Antoun, F. Baly, and H. Hajj, “AraBERT: Transformer-based model for Arabic language understanding,” in *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*. Marseille, France: European Language Resource Association, May 2020, pp. 9–15. [Online]. Available: <https://www.aclweb.org/anthology/2020.osact-1.2>
- [108] S. Eltanbouly, M. Bashendy, and T. Elsayed, “Simple but not naïve: Fine-grained Arabic dialect identification using only n-grams,” in *Proceedings of the Fourth Arabic Natural Language Processing Workshop*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 214–218. [Online]. Available: <https://aclanthology.org/W19-4624>

- [109] P. Přibáň and S. Taylor, “ZCU-NLP at MADAR 2019: Recognizing Arabic dialects,” in *Proceedings of the Fourth Arabic Natural Language Processing Workshop*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 208–213. [Online]. Available: <https://aclanthology.org/W19-4623>
- [110] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutiérrez, and K. J. Kochut, “A brief survey of text mining: Classification, clustering and extraction techniques,” *ArXiv*, vol. abs/1707.02919, 2017.
- [111] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, “A neural probabilistic language model,” *The journal of machine learning research*, vol. 3, pp. 1137–1155, 2003.
- [112] F. Almeida and G. Xexéo, “Word embeddings: A survey,” 2019.
- [113] R. Collobert and J. Weston, “A unified architecture for natural language processing: Deep neural networks with multitask learning,” in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 160–167.
- [114] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” *arXiv preprint arXiv:1310.4546*, 2013.
- [115] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

- [116] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching word vectors with subword information,” 2017.
- [117] J. Howard and S. Ruder, “Universal language model fine-tuning for text classification,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 328–339. [Online]. Available: <https://www.aclweb.org/anthology/P18-1031>
- [118] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: A robustly optimized bert pretraining approach,” *arXiv preprint arXiv:1907.11692*, 2019.
- [119] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, “Xlnet: Generalized autoregressive pretraining for language understanding,” *arXiv preprint arXiv:1906.08237*, 2019.
- [120] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “Albert: A lite bert for self-supervised learning of language representations,” *arXiv preprint arXiv:1909.11942*, 2019.
- [121] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *arXiv preprint arXiv:1910.10683*, 2019.

- [122] A. B. Soliman, K. Eissa, and S. R. El-Beltagy, “Aravec: A set of arabic word embedding models for use in arabic nlp,” *Procedia Computer Science*, vol. 117, pp. 256–265, 2017.
- [123] A. Erdmann, N. Zalmout, and N. Habash, “Addressing noise in multidialectal word embeddings,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2018, pp. 558–565.
- [124] M. A. Zahran, A. Magooda, A. Y. Mahgoub, H. Raafat, M. Rashwan, and A. Atyia, “Word representations in vector space and their applications for arabic,” in *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, 2015, pp. 430–443.
- [125] M. Abdul-Mageed, H. Alhuzali, and M. Elaraby, “You tweet what you speak: A city-level dataset of Arabic dialects,” in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA), May 2018. [Online]. Available: <https://www.aclweb.org/anthology/L18-1577>
- [126] A. Safaya, M. Abdullatif, and D. Yuret, “KUISAIL at SemEval-2020 task 12: BERT-CNN for offensive speech identification in social media,” in *Proceedings of the Fourteenth Workshop on Semantic Evaluation*. Barcelona (online): International Committee for Computational Linguistics, Dec. 2020, pp. 2054–2059. [Online]. Available: <https://www.aclweb.org/anthology/2020.semeval-1.271>

- [127] “Counting characters | docs | twitter developer platform,” <https://developer.twitter.com/en/docs/counting-characters#:~:text=In%20most%20cases%2C%20the%20text,as%20more%20than%20one%20character.,> (Accessed on 10/16/2021).
- [128] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, “Unsupervised cross-lingual representation learning at scale,” 2020.
- [129] “Api reference — tweepy 3.10.0 documentation,” <https://docs.tweepy.org/en/latest/api.html>, (Accessed on 03/18/2021).
- [130] “Gps coordinates for latitude: 23.48690, longitude: 44.82030,” <https://latitude.to/lat/23.48690/lng/44.82030>, (Accessed on 10/22/2021).
- [131] “”, <https://ar.mo3jam.com/>, (Accessed on 10/22/2021).
- [132] “Atlas,” <http://www.atlasalhajaat.com/>, (Accessed on 05/16/2021).
- [133] “Arabic stop words,” <http://www.abuelkhair.net/index.php/en/arabic/arabic-stop-words>, (Accessed on 10/22/2021).
- [134] I. A. El-Khair, “Effects of stop words elimination for arabic information retrieval: A comparative study,” 2017.
- [135] E. Clark and K. Araki, “Text normalization in social media: Progress, problems and applications for a pre-processing system of casual english,” *Procedia - Social and Behavioral Sciences*, vol. 27, pp. 2–11, 2011,

- computational Linguistics and Related Fields. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1877042811024049>
- [136] T. Baldwin, P. Cook, M. Lui, A. MacKinlay, and L. Wang, “How noisy social media text, how different social media sources?” in *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, 2013, pp. 356–364.
- [137] “(1) -”, <https://khamsat.com/>, (Accessed on 10/22/2021).
- [138] X. Tang, H. Mou, J. Liu, and X. Du, “Research on automatic labeling of imbalanced texts of customer complaints based on text enhancement and layer-by-layer semantic matching,” *Scientific Reports*, vol. 11, no. 1, pp. 1–11, 2021.
- [139] A. Radford and K. Narasimhan, “Improving language understanding by generative pre-training,” 2018.
- [140] R. Sennrich, B. Haddow, and A. Birch, “Neural machine translation of rare words with subword units,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725. [Online]. Available: <https://aclanthology.org/P16-1162>
- [141] “Lei mao’s log book – byte pair encoding,” <https://leimao.github.io/blog/Byte-Pair-Encoding/>, (Accessed on 11/01/2021).
- [142] M. Ott, S. Edunov, D. Grangier, and M. Auli, “Scaling neural machine translation,” in *Proceedings of the Third Conference on Machine Translation:*

- Research Papers*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 1–9. [Online]. Available: <https://aclanthology.org/W18-6301>
- [143] Y. You, J. Li, J. Hseu, X. Song, J. Demmel, and C.-J. Hsieh, “Reducing bert pre-training time from 3 days to 76 minutes,” *ArXiv*, vol. abs/1904.00962, 2019.
- [144] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, “Huggingface’s transformers: State-of-the-art natural language processing,” 2020.
- [145] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2017.
- [146] J. Pustejovsky and A. Stubbs, *Natural Language Annotation for Machine Learning*, ser. A Guide to corpus-building for applications. O’Reilly Media, Incorporated, 2012, no. v. 9, p. 878. [Online]. Available: <https://books.google.com.sa/books?id=QtzmqamXxx4C>
- [147] R. AlYami and R. AlZaidy, “Arabic dialect identification in social media,” in *2020 3rd International Conference on Computer Applications Information Security (ICCAIS)*, 2020, pp. 1–2.
- [148] W. Tok, A. Bahree, and S. Filipi, *Practical Weak Supervision: Doing More with Less Data*. O’Reilly Media, Incorporated, 2021. [Online]. Available: <https://books.google.com.sa/books?id=MjCHzQEACAAJ>

- [149] ———, *Practical Weak Supervision*. O'Reilly Media, 2021. [Online]. Available: <https://books.google.com.sa/books?id=mLpFEAAAQBAJ>
- [150] Z.-H. Zhou, “A brief introduction to weakly supervised learning,” *National science review*, vol. 5, no. 1, pp. 44–53, 2018.
- [151] J. E. Van Engelen and H. H. Hoos, “A survey on semi-supervised learning,” *Machine Learning*, vol. 109, no. 2, pp. 373–440, 2020.
- [152] O. Chapelle, B. Scholkopf, and A. Zien, *Semi-Supervised Learning*, ser. Adaptive Computation and Machine Learning series. MIT Press, 2010. [Online]. Available: <https://books.google.com.sa/books?id=A3ISEAAAQBAJ>
- [153] X. J. Zhu, “Semi-supervised learning literature survey,” 2005.
- [154] H. Scudder, “Probability of error of some adaptive pattern-recognition machines,” *IEEE Transactions on Information Theory*, vol. 11, no. 3, pp. 363–371, 1965.
- [155] S. Fralick, “Learning to recognize patterns without a teacher,” *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 57–64, 1967.
- [156] A. Agrawala, “Learning with a probabilistic teacher,” *IEEE Transactions on Information Theory*, vol. 16, no. 4, pp. 373–379, 1970.
- [157] X. Zhu and A. B. Goldberg, “Introduction to semi-supervised learning,” *Synthesis lectures on artificial intelligence and machine learning*, vol. 3, no. 1, pp. 1–130, 2009.

[158] “Google colab,” <https://research.google.com/colaboratory/faq.html>, (Accessed on 11/06/2021).

VITAE

- Name: Reem AlYami
- Nationality: Saudi Arabia
- Email: *reem.yami@kfupm.edu.sa*
- Permanent Address: Dammam, Eastern Province, Saudi Arabia
- Publications:
 - AlYami, R. and AlZaidy, R., 2020, March. Arabic dialect identification in social media. In 2020 3rd International Conference on Computer Applications Information Security (ICCAIS), IEEE.
 - AlYami, R. and AlZaidy, 2022, December. Weakly and Semi-Supervised Learning for Arabic Text Classification using Monodialectal Language Models. In 2022 Proceedings of the Seventh Arabic Natural Language Processing Workshop, Association for Computational Linguistics.
 - AlYami, R. and AlZaidy. A3D: A General Purpose Large-Scale Automatically Annotated Dataset for Arabic Dialect. [In submission]