

AMERICAN UNIVERSITY OF BEIRUT

Information Extraction from Arabic Social Media
Content

by
Rayan Dankar

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Engineering
to the Department of Electrical and Computer Engineering
of the Faculty of Engineering and Architecture
at the American University of Beirut

September 2020

AMERICAN UNIVERSITY OF BEIRUT

Information Extraction from Arabic Social Media Content

by
Rayan Dankar

Approved by:

Dr. Fadi Zaraket, Associate Professor
Electrical and Computer Engineering

Advisor



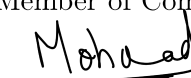
Dr. Ali Chehab, Professor
Electrical and Computer Engineering

Member of Committee



Dr. Mohamad Jaber, Associate Professor
Computer Science

Member of Committee



Date of thesis defense: September 8, 2019

AMERICAN UNIVERSITY OF BEIRUT

THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: Dankar Rayan Hilal
Last First Middle

☒ Master's Thesis ☐ Master's Project ☐ Doctoral Dissertation

☐ I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

☒ I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes after: **One** ___ year from the date of submission of my thesis, dissertation or project.
Two ___ years from the date of submission of my thesis , dissertation or project.
Three ___ years from the date of submission of my thesis , dissertation or project.

Rayan Sep 15th, 2020
Signature Date

This form is signed when submitting the thesis, dissertation, or project to the University Libraries

Acknowledgements

To begin with, I want to take the time to express my deepest gratitude and appreciation to my parents. Thank you mom and dad, Sana and Hilal, for always being there for me, for never standing in the way of my dreams and for always believing in me. Thank you for all the effort and sacrifices you put in to allow me to obtain my Master's degree. You are and will always be my support system. You are the wand that makes all my dreams come true.

I want to thank my thesis advisor, Prof. Fadi Zaraket, for making this happen. Thank you for your continuous support and mentorship throughout my graduate journey.

À ma grand-mère Najah, merci pour ton amour inconditionnel, ta générosité sans fin et ton coeur en or.

To my aunts, thank you for always being on my side. You are always there cheering me on and waiting on the sidelines to step in if I need any help. A special thank you to Fida for being a pillar in my education journey from my undergraduate to graduate degree.

To my sisters, thank you for being so understanding. Thank you for always reminding me that I am able to accomplish my dreams.

To my roommates and second family, thank you for being blindly supportive. To Haneen and Rawaa, you help me stand when I am at my worst and cheer me on when I am at my best.

To my friends, Alaa, Obeida and Wissam, thank you for your help, for listening to me vent, and for providing technical and emotional support.

To all my family and friends, to everyone who was a part of making this thesis happen, thank you!

An Abstract of the Thesis of

Rayan Dankar for Master of Engineering
Major: Electrical and Computer Engineering

Title: Information Extraction from Arabic Social Media Content

Stakeholders such as advertisers, celebrities and politicians are showing high interest in information extraction from social media content. Social media content includes posts and interactions in local dialects.

Arabic and its local dialects are among the top used languages in the world. Modern standard Arabic is dominant in formal platforms and events such as newscasts, speeches, books, and newspapers. Dialects of Arabic are dominant in everyday communication and are increasingly used on social media platforms.

Information extraction techniques focus on extracting entities and relational entities from unstructured text. They have limited support for MSA and lack support for Arabic dialects.

In this thesis, we construct necessary resources for information extraction from Arabic social media content. We introduce a method to retrieve relevant information by extracting entities and relational entities from modern standard and dialectical Arabic text. To improve entity and relational entity extraction, we construct ADAT, an Arabic Dialect Annotation Tool. ADAT serves as a building block to construct computational linguistic models from Arabic text and requires basic linguistic knowledge from its users.

We evaluate the obtained results from our work on entity and relational entity extraction on a corpora concerning Yemen and report its performance using precision and recall metrics. We finally use graph construction and analysis techniques to draw insights from the extracted entities and relational entities.

Contents

Acknowledgements	v
Abstract	vi
1 Introduction	1
1.1 Information Extraction	1
1.2 Computational Linguistic Resources	2
1.3 Aims	2
1.4 Framework	3
2 Literature Review	5
3 Motivating Examples	8
3.1 World Cup Example	8
3.2 Missing Journalist Example	13
4 Background	16
4.1 Arabic Morphology	16
4.1.1 Tokenization	16
4.1.2 Inflectional morphology	17
4.1.3 Diacritics	17
4.1.4 Linguistic Features	17
4.2 Entity and Relational Entity Extraction	18
4.3 Graph Extraction and Analysis	19
4.3.1 Graph Construction	19
4.3.2 Graph Analytics	19
5 ADAT: Arabic Dialect Annotation Tool	20
5.1 Annotator Work Flow	21
5.1.1 Annotation Task	22
5.1.2 Database	27

6	Social Media Data Collection	30
6.1	Tweet Extraction	30
6.2	Pre-processing	31
6.3	Data Set Construction for Entity and Relational Entity Tagging .	33
7	Methodology for Entity and Relational Entity Extraction	34
7.1	Entity and Relational Entity Matching	34
7.1.1	N-grams	34
7.1.2	Morphological Analysis	35
7.2	Distributional similarity	35
7.3	Word2Vec Modal	35
7.4	Vocabulary Builder	36
7.5	Similar Words	36
7.6	Manual Inspection Iterative Approach	38
7.7	Neural Networks	39
7.7.1	RNNs	39
7.7.2	LSTM	39
8	Leveraging Partial Tuples	41
8.1	Partial Tuple $(e1, e2, ?)$	41
8.2	Partial Tuple $(e1, ?, r)$	42
9	Results	44
9.1	Testing Data Set	44
9.2	Results Evaluation	44
9.2.1	Confusion Matrix	44
9.2.2	Evaluation Metrics	44
9.3	Initial data	46
9.4	Evaluation Results	46
9.5	Numeric Evaluation Results	48
9.5.1	Recall	48
9.5.2	Precision	48
9.5.3	Fscore	48
9.6	Evaluation Results Plot	49
9.6.1	Recall, Precision and F1score Plot	49
9.6.2	Unmatched Words Plot	49
10	Graph Analysis and Insights	51
10.1	Graph Data	51
10.2	Graph Construction	53
10.3	Graph Visualization Algorithms	53
10.3.1	Force Atlas Algorithm	54
10.3.2	Graph Rank Degree	55

10.3.3 Average Path Length and Betweenness Centrality	58
10.3.4 Community Detection	61
10.4 Graph Insights	62
11 Conclusion	65
A Abbreviations	66

List of Figures

1.1	Methodology Framework	3
3.1	Tweet Example	9
3.2	Tweet Example Analysis: extracting entities and relational entities	10
3.3	Tweet Example Analysis: linking entities and relational entities .	11
3.4	Graph Construction	12
3.5	Arabic Tweet Example	13
3.6	Arabic Tweet Example Analysis	13
3.7	Graph 2 Construction	15
5.1	ADAT login page	21
5.2	Annotation word selection	22
5.3	Context Selection	23
5.4	Suggested annotations	24
5.5	Annotation Zone	24
5.6	Annotation Stage: choosing a suggested solution	25
5.7	Conjugation Box	26
5.8	Comments and Referral	27
5.9	Full View of ADAT	28
5.10	ADAT: saving the annotation	29
6.1	Python Example Using Twitter API	31
6.2	Unclean Tweet Example	32
6.3	Tweet Data Collection Steps	32
7.1	Vector Representation For	37
7.2	Example showing the similarity between two words	37

7.3	Example for Finding Similar Words	38
7.4	Hidden State	39
7.5	RNN Diagram Obtained from [1]	40
8.1	Adding partial tuples: $(e1, e2, ?)$	42
8.2	Adding partial tuples: $(e1, ?, r)$	43
9.1	Confusion Matrix	45
9.2	Results plot showing the recall, precision and fscore for the E and RE extraction at each stage in our applied techniques	49
9.3	Unmatched Words Plot showing the number of unmatched words at each stage in our applied techniques	50
10.1	Selected Top Frequent Words	53
10.2	Graph Illustration	54
10.3	Graph Illustration After Force Atlas Algorithm with a Repulsion Strength of 10,000	57
10.4	Graph Illustration After Force Atlas Algorithm with a Repulsion Strength of 20,000	58
10.5	graph illustration after rank degree	59
10.6	Graph Illustration After Ranking the Node Size on Betweenness Centrality	60
10.7	Example of Graph Detecting and Highlighting of Communities . .	61
10.8	Graph Illustration After Colorizing Based on Community Detection, along with the Color Key	63
10.9	Final Resulting Graph	64

List of Tables

2.1	Previous Work Comparison on E and RE Extraction on Accuracy and Fscore	7
3.1	Extracted Entities and Relational Entities	9
3.2	Extracted Entities and Relational Entities	14
9.1	Initial Data	46
9.2	Evaluation Metrics Results	47
10.1	Nodes and Edges Table	52
10.2	Degree of Each Entity Node	56

Chapter 1

Introduction

1.1 Information Extraction

Information extraction (IE) is the process of extracting specific information from text sources. Information extraction is a powerful technique used in natural language processing (nlp) which enables parsing through any piece of text.

Information extraction extracts structured information from unstructured text. It automates the retrieval of specific information topics from different text content. IE makes it possible to pull information from social media, websites and documents. It helps in finding hidden gems of information.

IE techniques focus on extracting entities and extracting relational entities from unstructured text. Entities are recognized by belonging to any of those categories: people, locations, organizations, and dates. Extracting relations relies on extracting the relations between the occurring entities such as located in, employed by, part of, married to, ...

Entities are extracted from text using formulae that range over linguistic features and using words similar to entities extracted based on information and statistical similarity metrics. Similarly, the set of extracted entities is represented by $E = \{e_1, e_2, \dots, e_{|E|}\}$ where the relations are, specified as tuples, and expressed by $R = \{r_1, r_2, \dots, r_{|R|}\}$. A relation tuple is expressed as $r = \langle e_1, e_2, e_r \rangle$ where e_1 and $e_2 \in E$ are entities, and e_r denotes the relation entity label of $r \in R$.

An example illustrating the importance of information extraction is Cambridge Analytica. Cambridge Analytica[2] is a data analytics firm which provides to its customers intuition on the consumer behavior. It played an important role in Donald Trump's campaign by gathering data on social users and modelling this data. The model provided intuition on the users' opinions thus producing tailored advertisements to try and alter their opinion[3].

Stakeholders such as advertisers, politicians, companies, and researchers are showing high interest in information extraction from social media content. They are interested in automating the understanding social media content.

There has been a significant rise in social media use over the past years. In 2020, around 3.6 billion people used social media worldwide [4]. Social media users express their views and opinions to discuss topics of interest and trending topics.

A trending topic is a subject that experiences a rise in popularity on social media platforms for a period of time. It receives large public interest where the majority express their opinions and views over this popular event. Trending topics may relate to current, social and breaking news. Trending topics allow the identification of the most important topics across regions during a specific period of time. Arabic is an active language used in social media. Around 5 million Arabic users on Twitter were recorded by early 2014 [5]. The Arabic language is one of the top most spoken languages worldwide. It is one of the six official languages in the United Nations[6]. It is spoken by more than 422 million people in around 22 countries worldwide[7] and is the worship language of all Muslims.

Modern Standard Arabic (MSA) is the formal Arabic. It is the common form of Arabic understood by all native speakers. Arabic is used in formal platforms and events such as newscasts, speeches, books and newspapers.

Arabic dialects are variations of Arabic used in everyday communication and expression, and they vary across regions. For example, Levantine in Lebanon, Syria, Jordan and Palestine, Egyptian in Egypt, and Gulf in Kuwait, Bahrain, Qatar, UAE and Oman.

The set of codes of all human languages [8] include 30 entries for Arabic including languages that use the Arabic characters such as Tajik and Cypriot.

Dialectical Arabic (DA) such as Levantine, Gulf and Moroccan Arabic differ from MSA and from each other along several linguistic features.

1.2 Computational Linguistic Resources

IE relies heavily on determining and understanding the computational linguistics of the input text. DA differs from MSA and from each other on several linguistic features. Some of these features are alphabet, orthography, phonology, morphology, syntax, and semantics discussed thoroughly in chapter 4, section 4.1. Information extraction techniques have limited support for MSA and lack support for Arabic dialects. We discuss previous work on IE in chapter 2. Work on IE from MSA text exists[9, 10] with limited support. Work on IE from DA text is still lagging and needs improvement[11, 12, 13].

1.3 Aims

In this thesis we construct necessary resources for information extraction from Arabic social media content, we introduce a method to retrieve relevant informa-

tion from MSA and DA text by: extracting entities and relational entities from posts, we evaluate our obtained results on entity and relational entity extraction on a corpora concerning Yemen, we report the performance using precision and recall metrics, we use graph construction and analysis techniques to draw insights from the extracted entities and relational entities, in an attempt to answer stakeholder questions

1.4 Framework

In this work, we propose a methodology for for information extraction from Arabic social media content. We introduce a method to retrieve relevant information by extracting entities and relational entities from modern standard and dialectical Arabic text. We construct and present ADAT, an Arabic Dialect Annotation Tool, to improve entity and relational entity extraction. ADAT serves as a building block to construct computational linguistic models from Arabic text and requires basic linguistic knowledge from its users.

The methodology applied in this thesis requires the construction and the application of the following necessary resources illustrated in figure 1.1:

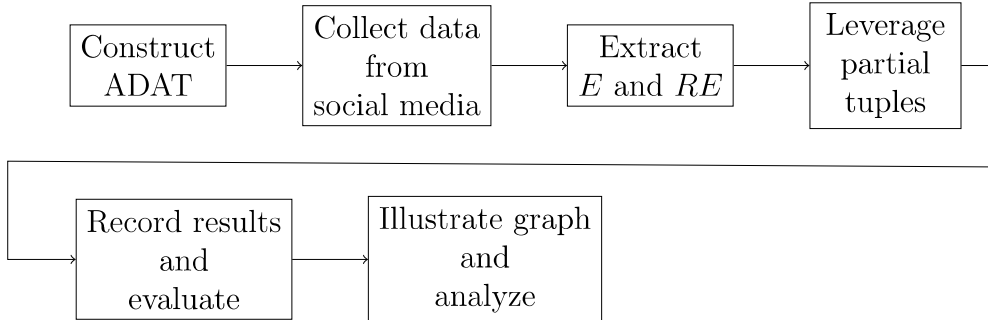


Figure 1.1: Methodology Framework

- To improve entity and relational entity extraction, we start by constructing ADAT, an Arabic Dialect Annotation Tool.
- We collect text data from social media.
- We extract entities and relational entities.
- We leverage partial tuple techniques to improve IE.
- We evaluate the obtained results from our work on entity and relational entity extraction on a corpora concerning Yemen.
- We report the performance using precision and recall metrics.

- We use graph construction and analysis techniques to draw insights from the extracted entities and relational entities.

In the rest of this thesis, we present some previous work on related topics in chapter 2. In chapter 3, we give a motivating example to illustrate the project's aim. We present necessary background material for some concepts in chapter 4. We present the tool ADAT in chapter 5. We discuss data collection and processing from social media in chapter 6. We set the methodology to be followed for entity and relational entity extraction in chapter 7. We work on leveraging some partial tuples in chapter 8. We analyze and plot our results in chapter 9. We illustrate, analyze, and discuss our resulting graph in chapter 10. Finally, we conclude this thesis in chapter 11.

Chapter 2

Literature Review

The objective of this literature review chapter is to give a general overview, about some important previously existing work, on some of the concepts related to the thesis work.

Previous work has been dedicated to Arabic morphology. The work in [11] introduced MADAR Arabic dialect corpus and lexicon. MADAR presents a framework for a number of different dialects with mutual annotation guidelines. They built MADAR corpus which covers Arabic dialects of some cities in addition to English, French and modern standard Arabic (MSA). They translated sentences from Basic Traveling Expression Corpus (BTEC) used for tourism according to a set of guidelines. The MADAR lexicon structure is organized around concept keys which are defined in triplet words: English, French, and MSA. Each concept has a number of words associated with it, where each word is defined in terms of CODA orthography, CAMEL phonology [12], and number of cities used. Lexicon concepts are then identified and manually validated. They also presented a lexicon resource of around 1000 entries in each city's dialect.

There has been past work targeting morphological analysis on Arabic in both its MSA and DA form. The work in [10] present YAMAMA, a multi dialect Arabic morphological analyzer and disambiguator which follows MADAMIRA. MADAMIRA [9] delivers an output with the following: diacritization, part-of-speech, tokenization, gloss, inflection features and lemmatization. YAMAMA is similar to MADAMIRA but differs in its disambiguation models which increases its speed. YAMAMA is almost five times faster than MADAMIRA. It delivers its output in the same format as MADAMIRA but with a richer representation. Yamama analyzes each word by using maximum likelihood for in-vocabulary words; it ranks the analyses for out-of-vocabulary words using lemma and Buckwalter POS tag unigram language models.

The work in [12] presents a set of guidelines for Arabic dialect orthography by introducing CODA*, an extension of CODA [14], Conventional Orthography for DA. CODA* guidelines add clarifications on the exception list of CODA. CODA* introduces a seed lexicon containing examples on dialect words. The

seed lexicon increases with the increase of the dialect thus becoming a dictionary of the dialect. They also set a list of phonology and specification rules for CODA* construction. They introduce as well a phonological representation to compare the dialect lexicons.

The work in [13] presents Curras which is the first morphological corpus of Palestinian Arabic dialect (PAL). They extend CODA guidelines to include specifics of PAL where they add phonology, morphology and a word list of exceptional words for PAL. They then add part-of-speech, stem, prefix, suffix, lemma and gloss as annotations to Curras.

There are some existing annotation tools that support Arabic. BRAT [15] and WorkFreak [16] are web annotators which permit building entity and relational entity corpora. MMAX2 [17] is an XML based annotation tool. DIWAN is a desktop application which can also be used online for dialect annotation. The Quranic Arabic Corpus [18] presents a morphological annotated corpus for the Holy Quran. BAAC [19] is an annotated Arabic corpus built by using the annotator they developed. However, BAAC is built for MSA.

ADAT is different from all these tools by allowing the annotation of DA and MSA. It is not restricted to a certain dialect and allows the annotation of any dialect.

Some previous work is dedicated to Information Extraction. Information extraction is being able to extract structured information from the unstructured information posted. IE performs analysis on text to retrieve information on entities and relationships.

In table 2.1, we display the comparison of some selected previous work and discuss them in what follows.

In their work, Abuzayed et al. [20], present a quick and simple approach for detecting hate speech in Arabic tweets. They rely on classical and neural learning modals and result in a 73% Fscore. Although they targeted MSA and DA, they only classified the tweet text into either hate speech or not hate speech.

In their work, Taghizadeh et al. [21], present a cross language method to extract relational entities from text. They score a 63.5% fscore. However, they only target MSA and work on RE.

There is the work of Zaraket et al. [22] which presents ANGE, a graph extraction method for hadith and biography literature. They use cross-document NLP to improve the accuracy of entity and relation extraction. They result in a 93% accuracy score; however, they only target MSA.

In the work of Jay [23], a user-driven relational model is implemented for entity and relational entity extraction. They result in an fscore of 50% for entities and of 72% for relational entities; however, they target MSA in newspapers.

There exists more related work in IE of which we select and discuss in the following. In their work, Habib et al. [24] present a framework for IE from the unstructured text posted on social media. They discuss the challenges faced in information extraction from social media and how to overcome them using their

framework. Their components include noisy text filtering to collect relevant text based on domain or language, named entity extraction without relying on POS and capitalization, named entity disambiguation by linking users to their social network page, feedback loops and uncertainty handling.

In his work, Traboulsi [25] discusses how a local grammar approach can be used to extract Arabic person names as entities from Arabic text. Jaber et al. present MERF [26], a framework for Arabic morphology-based entity and relational entity extraction. MERF provides a user interface requiring the association between tag types defined by the user and regular expressions over Boolean formulae. It constructs entities and relational entities after matching the user specifications such as tag types and morphological features.

In their work, Zhang et al. [27] explain how cross- document studies relations between terms within some documents. This information is important for information extraction and retrieval from documents.

Work	Arabic	Accuracy	Fscore	Features	Issues
Quick and simple approach for detecting hate speech in arabic tweets [20]	MSA DA		73%	classical and neural learning models	classification of hate speech
Cross-Language Learning for Arabic Relation Extraction [21]	MSA		63.50%	cross language method for RE	MSA, RE
Arabic Cross-Document NLP for the Hadith and Biography Literature [22]	MSA	93%		cross-document NLP	MSA hadith biography
User-driven relational models for entity-relation search and extraction [23]	MSA		E: 50% RE:72%	user-driven relational models	MSA newspapers

Table 2.1: Previous Work Comparison on E and RE Extraction on Accuracy and Fscore

We record our results in chapter 9 and discuss and analyze them thoroughly.

Chapter 3

Motivating Examples

In this chapter, we give some motivating examples to illustrate the outcomes of this thesis work. We show how, starting with some social media content, we are able to extract its entities and relational entities. We then show how we are also able to construct the graph representing the social media content.

3.1 World Cup Example

Figure 3.1 shows a tweet posted on Twitter platform and a reply to the tweeted post.

In this tweet, Dr. Fadlo Khuri, is expressing supporting Germany in the World Cup. He states that its manager, Joachim Low, needs to step down. He also states that if its manager doesn't step down, France may score 10 points. A twitter user, Wael Darwish, replies to Dr. Fadlo Khuri, agreeing with him and adding that the manager, Joachim Low, should be replaced by Jurgen Klopp.

We will extract the entities and relational entities present in the discussed tweet to construct its respective graph.



Figure 3.1: Tweet Example

Figure 3.2 shows the entities and the relational entities that will be extracted from the tweet. The entities are the words underlined and the relational entities are the words outlined by rectangles.

The entities and relational entities that are extracted from this tweet are recorded in tables 3.1a and 3.1 respectively.

Entities		
person names	country names	objects
Fadlo Khuri	Germany	World Cup
Wael Darwish	France	Confederation
Jeurgon Klopp		
Joachim Low		

(a) Extracted Entities

Relations	
verbs	adjectives
follow	winning
step down	ponderous
score 10	predictable
replace by	clueless

(b) Extracted Relational Entities

Table 3.1 Extracted Entities and Relational Entities



Figure 3.2: Tweet Example Analysis: extracting entities and relational entities

The entities extracted are: Fadlo, Germany, Wael, Klopp, Joachim Low, World Cup, Confederation, and France. The entities will construct the nodes in the graph.

The extracted relational entities are: follow, support, winning, step down, ponderous, score 10, and replace by. The relational entities will construct the edges connecting the nodes in the graph.

Figure 3.3 shows how the extracted entities and the extracted relational entities from the tweet are linked together.

After extracting the entities and relational entities, we will be able to construct their respective graph representation. Figure 3.4 shows the final formed constructed graph.



Figure 3.3: Tweet Example Analysis: linking entities and relational entities



Figure 3.4: Graph Construction

3.2 Missing Journalist Example

In this example, the Americans for Democracy & Human Rights in Bahrain (ADHRB) post a tweet in Arabic language. The tweet discusses the latest updates on the case of the detainment of the reporter, Jamal Khashoggi, in the Saudi Consulate in Istanbul.

Figure 3.5 shows this Arabic tweet.



Figure 3.5: Arabic Tweet Example

Figure 3.6 shows the entities and the relational entities that will be extracted from the tweet. The entities are the words underlined in the tweet and the relational entities are the words outlined in rectangles.



Figure 3.6: Arabic Tweet Example Analysis

The entities and relational entities extracted from this tweet are displayed in table 3.2a and table 3.2b respectively. The entities extracted are: ADHRB,

Entities		
person names	places	dates
جمال خاشقجي Jamal Khashoggi	قنصلية consulate	أكتوبر 2 2018 October 2 2018
ADHRB	سعودية Saudi Arabia	

(a) Extracted Entities

Relational Entities
مستجدات <i>mstġdāt</i> : updates
احتجاز <i>āhtġāz</i> : detainment
إخفاؤه <i>ihfāwh</i> : conceal
تضع <i>tḍ</i> : put

(b) Extracted Relational Entities

Table 3.2 Extracted Entities and Relational Entities

جمال خاشقجي *ġmāl ḥāšqġy* , قنصلية *qnṣlyh* , السعودية *sʿwdyh* ,

أكتوبر 2 2018 *ʾaktwbr* . The entities will construct the nodes in the graph.

The relational entities are: مستجدات *mstġdāt* , احتجاز *āhtġāz* , إخفاؤه *ihfāwh* , تضع *tḍ* . The relational entities will construct the edges connecting the nodes in the graph.

Figure 3.7 shows the final constructed graph formed by the extracted entities and relational entities.

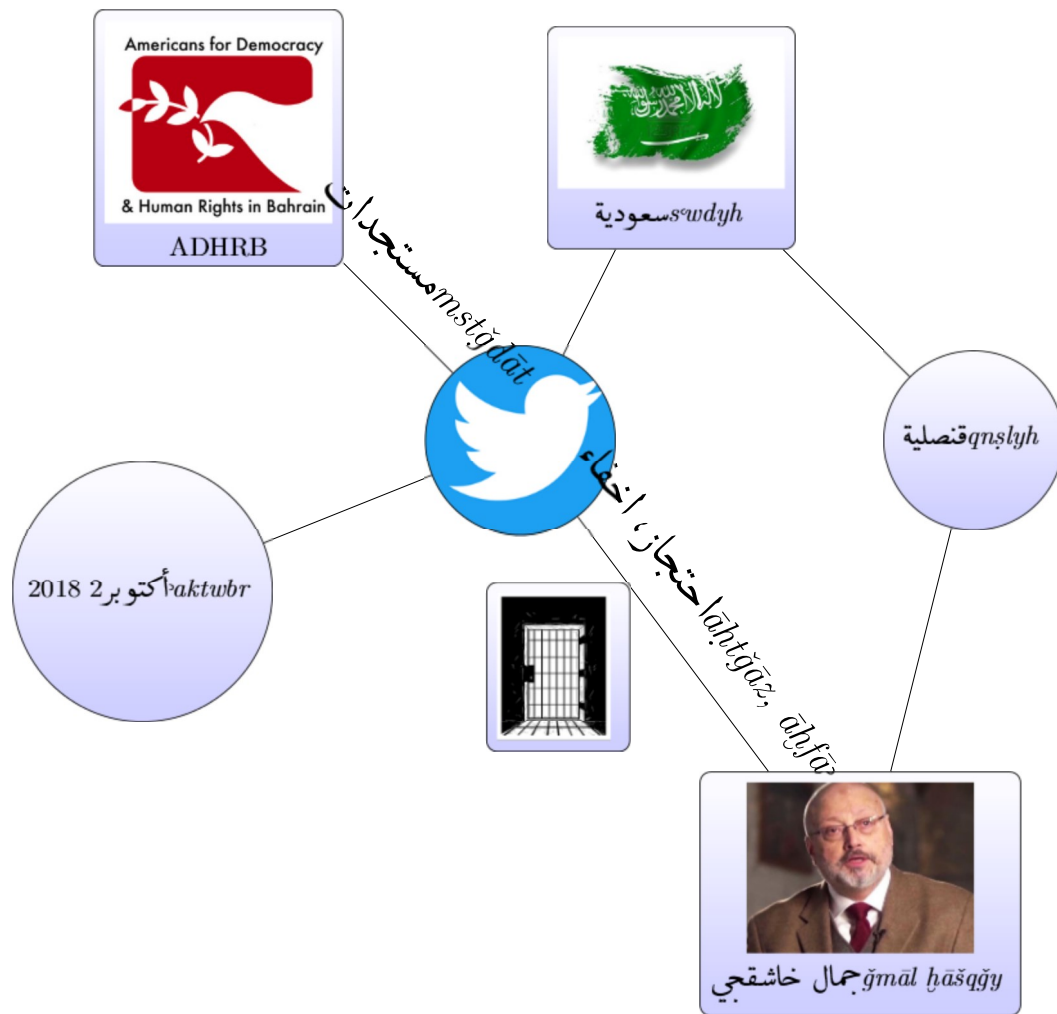


Figure 3.7: Graph 2 Construction

Chapter 4

Background

The social media phenomenon has increased in volume with the emergence of social media platforms such as Facebook[28], Twitter[29], Instagram[30] and LinkedIn[31].

Natural language processing methods enable information extraction. There is an increased interest for information extraction lately.

For this work, we extract meaningful information from tweets. Tweets are text posts or messages posted to Twitter platform. We will collect tweets, extract *entities* from the tweets, extract *relations* between the extracted entities, and analyze the entities and the relations using *graph* construction and analysis techniques to answer *questions of interest*. In what follows, in this chapter, we provide necessary background on each of the following important concepts:

- Arabic morphology
- Entity and relational entity extraction
- Graph extraction and analysis

4.1 Arabic Morphology

Arabic language is morphologically rich and complex. It includes inflectional morphology to express gender, tenses and number. The morphological richness of Arabic language requires morphological analysis at different levels such as tokenization and clitics.

4.1.1 Tokenization

Morphological analysis is required for tokenization where white space separators between words can be omitted. This is due to the Arabic letters changing forms according to their position in the word (beginning, middle, end). This allows the

reader to separate the words visually rather than relying on the space separation. For example:

حلواشيه *hlwāshyh* which means delicious dessert, is a two word fragment: حلوا *hlwā* and شيه *shyh* .

4.1.2 Inflectional morphology

The rich inflectional morphology of Arabic is expressed in its attachable clitics. Prefixes and suffixes can attach to the word at the same time. The word can have several prefixes and several suffixes attached to it at the same time. An inflected word forms a complete sentence on its own. In the example وسيرميها *wsyrmyhā* , we observe only one word, however; it means: and he will throw it. The single word وسيرميها *wsyrmyhā* , has the attached clitics: ها *hā* , س *s* , و *w* making the word inflected. This inflected word forms a whole sentence on its own.

4.1.3 Diacritics

Arabic language has optional diacritics. Diacritics, though optional, sometimes change the tense, part of speech or even the whole meaning of the word. The word عمل *ml* is an example where adding diacritics to the word changes its tense or its part of speech. عَمِلَ *amalun* means the noun work, عَمِلَ *umila* means was done, and عَمِلَ *amila* means worked.

Another example is the word جزر *ǧzr* , where specifying the diacritics, changes the whole meaning of the word. جَزَرَ *ǧzr* with diacritics can be interpreted as جُزُرُ *ǧuzur* meaning islands or as جَزَرُ *ǧazar* meaning carrots.

4.1.4 Linguistic Features

Arabic dialects differ from MSA and from each other on several linguistic features. Some of these linguistic features are listed as follows:

- Alphabet:

- The letter ف *v* represents the phoneme /v/ in loanwords such as the

word فيينا *vyynā* (Vienna)

- Phonology:

- the word حقوق *ḥqwq* (rights) sounds حوءء *ḥwʿw* in Lebanese

- Morphology:

- the negation of the word بدي *bdy* (I want) can be either the word بديش *bdyš* or the word ما بدي *mā bdy*

- Orthography:

- The word المجيد *ālmğyd* can have its letters written in a different format

- Semantics

- The word الليمون *lymwn* means oranges in Lebanese and means lemons in Gulf and Egyptian

4.2 Entity and Relational Entity Extraction

In this work, we use computational models to perform entity extraction and relational entity extraction.

An entity $E = \{e_1, e_2, \dots, e_{|E|}\}$ is a term which distinguishes between several items that have similar attributes. Entity examples are names, locations, and objects. Relational entity $R = \{r_1, r_2, \dots, r_{|R|}\}$ shows the relation linking between two different extracted entities.

A relational entity is a tuple $r = \langle e_1, e_2, e_r \rangle$ where:

- e_1 and $e_2 \in E$
- e_1 and e_2 are entities
- e_r denotes the relation entity label of $r \in R$

4.3 Graph Extraction and Analysis

A graph is a non linear data structure constituted of nodes and edges. Edges connect the nodes to each other.

Graphs can be either directed or undirected. A directed graph is a graph where the edges are ordered pairs of nodes such that: an edge (u, v) connecting node u to node v , is different than an edge (v, u) connecting node v to node u , $(u, v) \neq (v, u)$. An undirected graph is a graph where $(u, v) = (v, u)$

4.3.1 Graph Construction

A graph $G = (V, E)$ is a graph where V represents a set of vertices or nodes and E represents a set of edges.

- The entities E are represented by the vertices V .
- The relational entities R are represented by the edges E .

4.3.2 Graph Analytics

1. **cross-reference:** concerns identifying equivalent graph G elements:
 - $v_1 = v_2$ where $v_1, v_2 \in V$
 - $e_1 = e_2$ where $e_1, e_2 \in E$
2. **reachability:** where an entity can reach another entity by a defined relation
3. **important nodes:** where central nodes mean their respective entities are of high degree
4. **hierarchy:** which is displayed by strongly connected components.
5. **finding network characteristics:** where subnetworks are dense but the whole graph is not.

Chapter 5

ADAT: Arabic Dialect Annotation Tool

Due to the rich morphological state of Arabic language, we need an annotation tool resource to produce the necessary linguistic features for our text. Recognizing the linguistic features in return helps in the recognition of the text to be labelled as entities and relational entities which is at the heart of our aim, information extraction. We need an annotation tool that will surpass the previous existing tools discussed in chapter 2.

We construct ADAT, أداة *adāh*, an Arabic Dialect Annotation Tool. ADAT serves as a building block to construct computational linguistic models from Arabic text and requires basic linguistic knowledge from its users. Its importance is expressed in its ability to morphologically annotate data. The tool takes text as input and produces linguistic features as output. This allows us in determining whether the input word is an *entity* or a *relational entity*. ADAT surpasses the previous tools by providing the following:

- ADAT is a user friendly tool.
- ADAT integrates the use of morphological analyzers.
- ADAT targets annotating DA and MSA words.
- ADAT automatically suggests annotations to help the user annotate.

- ADAT annotates all the words which have the same meaning, even if present in different sentences, together.

In this chapter, we discuss how ADAT works and present all its features.

5.1 Annotator Work Flow

ADAT is an online platform where users register to an annotation task. Each user has an account and logs in to be able begin.

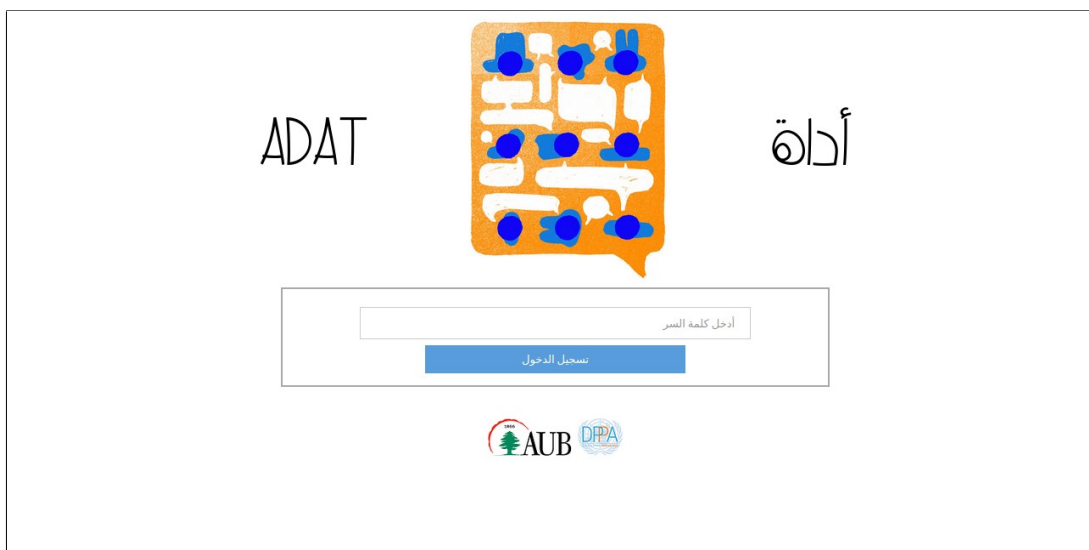


Figure 5.1: ADAT login page

Once the user logs in with his correct credentials, he will be redirected to the main page of ADAT. ADAT presents sentences containing the words that the user has to annotate. The user selects a word w , and all the sentences containing the word w are displayed. Each sentence defines a certain context of w . In the following, we describe how the annotation process works.

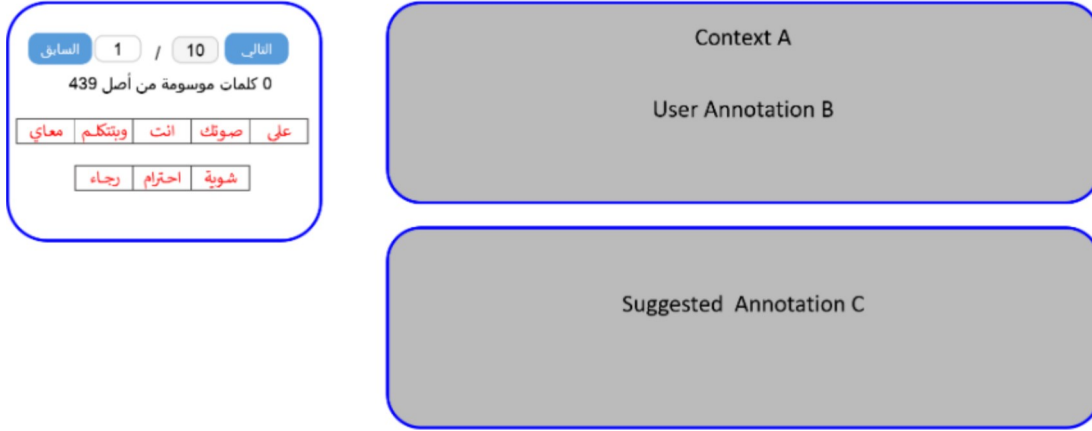


Figure 5.2: Annotation word selection

5.1.1 Annotation Task

Once the user logs in, ADAT displays a table containing the first sentence, that needs to be annotated. In Figure 5.2, the first sentence is *على صوتك انت وبتكلم معاي شوية احترام رجاء* *ly šwtk ānt wbttklm m'āy šwyh āḥtrām rǧā*. Each word in the sentence needs to be annotated. If the word is not annotated yet, it is displayed in a red color. If the word is annotated at least once before, it is displayed in a green color. The user can navigate back and forth to display the previous or the next sentence. In Figure 5.2, all the words of the displayed sentence are not annotated yet. The user has annotated 0 words out of 439 words.

The user clicks any word he wants to annotate to start the annotation process. Once a word w is clicked, all contextual sentences containing w are displayed in zone A. The user's annotation will be displayed in zone B and the suggested annotations will be displayed in zone C as further explained in what follows.

In Figure 5.3, we show all contextual sentences containing the clicked word w . The sentences, the sentence number, and a respective checkbox for each sentence are shown in zone A. w is highlighted in all the sentences for easier access.

The same word can have different meanings or share the same meaning between multiple sentences. The user has the option to select more than one sentence which share the same meaning for w to annotate them simultaneously. The user can annotate w in all the selected sentences at the same time. This reduces the number of non-annotated data. It also annotates a larger amount of data in less time and effort.

In Figure 5.2, the user selected the word *علي* *ly*, raise, which is not annotated before.

علي
اختر الجمل التي تحتوي نفس التصريفات والمعنى لكلمة:

علي صوتك انت وبتكلم معاي شوية احترام رجاء	1	<input type="checkbox"/>
سلم معاك علي علي وسمير	5	<input type="checkbox"/>

User Annotation B

Figure 5.3: Context Selection

Upon choosing a context, the tool displays a table with all the suggested morphological solutions for the chosen word shown in Figure 5.4. These suggestions will be displayed in zone C under *الحلول الصرفية* *ālhlwl ālsrfyh*, (morphological solutions). The suggested solutions are either generated by the morphological analyzer *SARF* [32] or fetched from the database from previous user annotations.

In our example, the case of the word *علي* *ly*, all the words are suggested from *SARF*, as there are no previous annotated solutions, by any user, for this word in the database yet.

We display the following in the morphological suggested solutions zone:

- source: is either *SARF* or a user's ID
- أصل الكلمة *asl ālklmh* : lemma
- الجذر *ālḡdr* : stem
- المعنى *ālmnā* : word meaning
- بادئة *bādʿyh* : prefix
- لاحقة *lāḡqh* : suffix
- التصريف *āltʿryf* : conjugation

In case none of the annotations are relevant, we provide a list of suggested

similar words to w under $كلمات مشابهة klmāt mšābhā$, (similar words). The user can pick a similar word to w , if he finds that none of the suggested annotations are relevant. The user can choose between these suggested words to display the suggested solutions for any one of them.

The words yly , ly , and wly are displayed as suggested similar solutions for ly .

We also set a search engine in this zone. The engine retrieves the morphological solutions of the word searched for.

الكلمة موضع الدرس: علي

كلمات مشابهة: علي , علي , وعلى

ابحث

الحلول الصرفية للكلمة: علي

Source	أصل الكلمة	الجزر	المعنى	بادئة	لاحقة	التصريف
SARF	عَلِيّ	عَلِيّ	Ali			عَلِيّ/اسم العلم
SARF	عَلِيّ	عَلِيّ	possessive form of Name of Person			
SARF	عَلِيّ	عَلِيّ	supreme/high			عَلِيّ/صفة
SARF	عَلِيّ	عَلِيّ	on/above			عَلِيّ/حرف جر+/مضاف الى مفرد متكلم
SARF	علي	علي	Name of Person			

Figure 5.4: Suggested annotations

At the annotation stage displayed in Figure 5.5, the user can: select and approve a suggested annotation from the list of suggestions, edit a suggested annotation, or enter a new annotation.

حفظ

تصريف كلمة علي:

اللهجة	بادئة	الجزر	لاحقة	أصل الكلمة	التصريف	المعنى
يمينية	None		None		+	

ملاحظة

درجّة التأكد:

إحالة الى:

Figure 5.5: Annotation Zone

Figure 5.6, shows the case of choosing a suggested annotation. The user clicks on the solution, and the annotation box in zone B is automatically filled with the respective chosen parameters. The user can edit any parameter by clicking on its respective cell.

حفظ تصريف كلمة علي: علي

اللهجة	بادنه	الجذر	لاحقه	أصل الكلمة	التصريف	المعنى
يعينية	None	علي	None	علي	علي/اسم العلم	Ali

ملاحظة

إحالة الى: درجة التأكد:

الكلمة موضع الدرس: علي

كلمات مشابهة:

يعلي علي وعلى

ابحث

الحلول الصرفية لكلمة: علي

Source	أصل الكلمة	الجذر	المعنى	بادنه	لاحقه	التصريف
SARF	علي	علي	علي			علي/اسم العلم
SARF	علي	علي	possessive form of Name of Person			
SARF	علي	علي	supreme/high			علي/صفة
SARF	علي	علي	on/above			علي/حرف جر + مضاف الى مفرد متكلم
SARF	علي	علي	Name of Person			

Figure 5.6: Annotation Stage: choosing a suggested solution

In case the user wants to enter a completely new annotation, the user needs to fill out the following features:

- dialect name
- prefix
- stem
- suffix
- lemma
- conjugation
- meaning

These features are shown in the table in zone B.

The dialect name, *اللهجة āllhgh*, is a drop down where the user chooses the dialect of the clicked word. The *بادنه bādbyh* is the prefix of the word and *لاحقه ā-hqh* is the suffix of the word. *الجذر ālgdr* is the clicked word's respective stem. *أصل الكلمة āsl āklmh* is the lemma of the clicked word. *المعنى ālmnā* is the meaning of the clicked word in English. *تصريف الكلمة tsryf āklmh* is the

conjugation of the word. *تصريف الكلمة tsryf āklmh*, displays a conjugation table, upon pressing the plus button, to facilitate specifying the conjugation. The user can not add new conjugation tags. Figure 5.7, shows the possible conjugation parameters to be chosen by the user. We list the options for each conjugation parameter:

- POS: *اسم āsm* (noun), *فعل fl* (verb), *ظرف zrf* (preposition), *ضمير dmyr* (pronoun)
- Number: *مفرد mfrd* (singular), *مثنى mtnā* (dual), *جمع ġm* (plural)
- Voice: *معلوم mʿlum* (passive), *مجهول mġhwl* (active)
- Gender: *مذكر mdkr* (male), *أنثى antā* (female), *محايد mhāyd* (neutral)
- Tense: *ماض māḍ* (past), *مضارع mḍār* (present), *مستقبل mstqbl* (future), *أمر amr* (order)

The user presses *نعم nm* (yes) and the conjugation will be filled into the conjugation field in the annotation zone.

POS	Number	Voice	Gender	Tense
اسم	مفرد	معلوم	مذكر	ماض
فعل	مثنى	مجهول	أنثى	مضارع
ظرف	جمع		محايد	مستقبل
ضمير				أمر

نعم

Figure 5.7: Conjugation Box

We provide a comments section for the user to enter any additional information under *ملاحظة mlāḥẓh* section. We also provide the user with a level of certainty selection option, as displayed in figure 5.8, to his annotation, in the *درجة التأكد drġh āltakd* section. The options are *جدا متأكد mtakd ġdā* (very

certain), *متأكد* *mtakd* (certain), and *مختار* *mhtār* (uncertain).

The user can also refer the word for further revision by checking the refer to, *إحالة إلى* *āḥālḥ ālā*, checkbox. This will allow him to choose between a list of options, he would like to refer his annotation to.

The screenshot shows the ADAT interface. At the top right, there is a blue button labeled 'حفظ' (Save) and a label 'تصريف كلمة على:' (Conjugate word on:). Below this is a table with the following columns: 'اللهجة' (Dialect), 'بادئة' (Prefix), 'الجذر' (Root), 'لاحقة' (Suffix), 'أصل الكلمة' (Original word), 'التصريف' (Conjugation), and 'المعنى' (Meaning). The table contains one row with the following values: 'يمنية' (Yemeni), 'None', 'عَلِيّ' (Ali), 'None', 'عَلِيّ' (Ali), 'عَلِيّ/اسم العلم' (Ali/Scientific name), and 'Ali'. Below the table, there is a section labeled 'ملاحظة' (Note) with a checkbox 'إحالة إلى:' (Refer to:) which is checked. Next to it is a dropdown menu labeled 'درجة التأكد' (Degree of certainty) with options: 'درجة التأكد' (Degree of certainty), 'متأكد جداً' (Very certain), 'متأكد' (Certain), and 'مختار' (Uncertain). There is also a button labeled 'المنسق' (Editor) and a text input field.

Figure 5.8: Comments and Referral

Figure 5.9 shows a full view of ADAT after specifying and filling in all the fields.

Once all the fields are specified, the user saves his work by pressing the *حفظ* *ḥfẓ*, save button as shown in figure 5.10. After saving, the word is either colored in green to indicate its complete annotation or it is colored in orange to indicate that the word's annotation is referred to someone and needs further revision.

5.1.2 Database

The database stores the data with its morphological attributes. The database contains a list of all the words in a selected number of sentences. It stores their respective morphological analysis: part of speech (POS), lemmas, dialect of the lemma, stem and English meaning.



أداة ADAT



السابق 1 / 10 التالي
 0 كلمات موسومة من أصل 439
 على صوتك انت وبتكلم معاي
 شوية احترام رجاء

اختر الجمل التي تحتوي نفس التصريفات والمعنى لكلمة: **علي**
 1 ☐ **علي** صوتك انت وبتكلم معاي شوية احترام رجاء
 5 ☐ سلم معاك **علي** وسمير
 حفظ تصريف كلمة **علي**:

المعنى	التصريف	أصل الكلمة	لاحقة	الجذر	بادئة	اللهجة
علي	غليّن/اسم العلم	غليّن	None	غليّن	None	بيانية

 درجة التأكد: حالة الى:

الكلمة موضع الدرس: **علي**
 كلمات مشابهة: **علي** **وعلي**
 البحث
 التحول للصيغة لكلمة: **علي**

Source	أصل الكلمة	الجذر	المعنى	بادئة	لاحقة	التصريف
SARF	غليّن	غليّن	possessive form of Name of Person			غليّن/اسم العلم
SARF	غليّن	غليّن	supreme/high			غليّن/صفة
SARF	غليّن	غليّن	on/above			غليّن/حرف جر/مضاف الى مفرد متكلم
SARF	غليّن	غليّن	Name of Person			غليّن/مضاف الى مفرد متكلم

Figure 5.9: Full View of ADAT



أداة ADAT



التالي 10 / 1 السابق

0 كلمات موسومة من أصل 439

علي صوتك انت وبتكلم معاي

شوية احترام رجاء

اختر الجمل التي تحتوي نفس التصريفات والمعنى لكلمة: علي

1 علي صوتك انت وبتكلم معاي شوية احترام رجاء

5 سلم معاك علي علي وسمير

لقد تم اضافة هذا الحل

OK

الكلمة موضع الدرس: علي

كلمات مشابهة: علي، وعلى

أبحث

التحليل الصرفي لكلمة: علي

Source	أصل الكلمة	الحذر	المعنى	بارنة	لاحقة	التصريف
SARF	علي	علي	possessive form of Name of Person			علي/صيغة
SARF	علي	علي	supreme/high			علي/صيغة
SARF	علي	علي	on/above			علي/صيغة
SARF	علي	علي	Name of Person			علي/صيغة

Figure 5.10: ADAT: saving the annotation

Chapter 6

Social Media Data Collection

In this chapter, we discuss how we construct our data set. We want to gather social media data to perform information extraction on its text. As such, we collect our data from Twitter platform to construct our data set. The reason, for choosing Twitter platform, lies in the fact that, Twitter provides a public API which allows a registered developer user to access the data, users have previously shared and posted on its platform.

6.1 Tweet Extraction

To extract our tweets, we follow the listed steps:

- Retrieval of Twitter API keys
- Twitter Connection for data extraction
- Saving the extracted data

Twitter provides a set of API endpoints for developers interested in using Twitter as a development platform.

To access Twitter API, we need an API key, an API secret, an Access Token, and an Access token secret. The key, token and secrets come with a twitter developer account. Twitter requires a detailed description of what the developer intends to do with the Twitter data before approving the use of its API.

To achieve Twitter Connection to extract tweets, we use Tweepy library[33]. Using Tweepy, we are able to connect to Twitter and to download the desired tweets. We extract the tweets according to different criteria such as: a specific keyword preceded by a hash-tag, or a specific user-name. Tweets can also be extracted according to a city's geolocation where each city has a unique geographic (GEO ID). GEO IDs are numeric codes used to identify each geographic area.

The following is a python example displayed in Figure 6.1 to extract tweets with the hash-tag peace.

```

import tweepy

ckey = 'xxx'
csecret = 'xxx'
atoken = 'xxx'
asecret = 'xxx'

auth = tweepy.OAuthHandler(ckey, csecret)
auth.set_access_token(atoken, asecret)
api = tweepy.API(auth)

maxTweets=100
for tweet in tweepy.Cursor(api.search,q='#peace',
count=100,lang="en",since="2018-10-03").items(maxTweets):

    print(tweet.text)

```

Figure 6.1: Python Example Using Twitter API

For this thesis, we extract Yemeni tweets. This thesis is in collaboration with a UN [34] project revolving around Yemen. As such, our tweets are either discussing Yemen related topics or posted in Yemen. Once we extract the tweets, we save them for processing.

We construct our data set by collecting around 40,000 tweets from different Yemeni Twitter accounts.

6.2 Pre-processing

Pre-processing is of utmost importance. Pre-processing is cleaning the raw extracted data. It is a preliminary step to having meaningful data. In section 4.1, we discussed Arabic morphology which factors in the importance of pre-processing. In addition to that, social media content contains unclean text and a lot of noise. Users express themselves, their opinion and emotions, on social media, by posting text containing special characters, extra punctuation marks and emojis, and repeated characters. A social media text post is full of clutter. Figure 6.2 shows an example of an unclean tweet. Pre-processing without lemmatization results

in the following sentence: جميل ربنا المباره $\dot{g}myl\ rb\dot{h}n\bar{a}\ \bar{a}lmb\bar{a}r\bar{a}h$, nice we won the game. After lemmatization, the sentence becomes: جميل ربح مباره $\dot{g}myl\ rb\dot{h}\ mb\bar{a}r\bar{a}h$, nice win game. Applying thorough pre-processing increases the accuracy

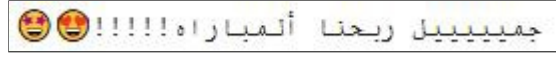


Figure 6.2: Unclean Tweet Example

of NLP tasks. In our work, the major steps of pre-processing are: normalization, noise removal, and lemmatization.

Figure 6.3, shows the steps needed to prepare our data set for information extraction.

The pre-processing is achieved in the following order:

- We normalize all instances of the letter ألف $\dot{a}lf$: $\dot{a}i$, $\dot{a}a$, \dot{a} , $\dot{a}\bar{a}$, $\dot{a}i$ to \dot{a} .
- We remove all diacritics.
- We remove extra white spaces.
- We remove all emojis.
- We remove all special non Arabic characters.
- We remove all repeating characters.
- We lemmatize all our words.

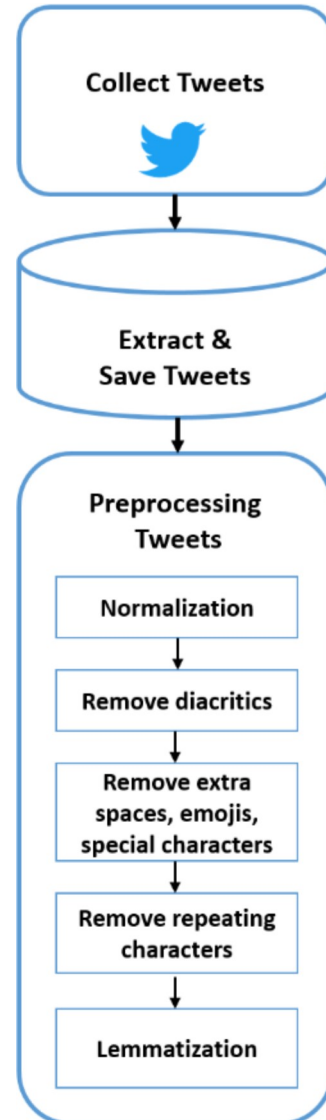


Figure 6.3: Tweet Data Collection Steps

6.3 Data Set Construction for Entity and Relational Entity Tagging

We start by specifying the bag of words that dictate the set of entities and relational entities for our Yemeni tweets. We then collect the synonyms for each of those words to expand our data set [35] and cover a bigger word range. We combine our findings thus constructing our annotation data set. We then finalize our corpus by lemmatizing our data set.

Chapter 7

Methodology for Entity and Relational Entity Extraction

In this chapter, we discuss all the techniques we follow in our methodology. The methodology applied is dedicated to be able to extract entities and relational entities from the processed tweet text which is at the core of this thesis work, information extraction.

7.1 Entity and Relational Entity Matching

We want to extract all the words in our collected cleaned tweets, section 6.2, that form entities and all the words that form relational entities. We start by matching our tweet words to the bag of words dictating our entities and to the bag of words dictating our relational entities, formed in section 6.3.

7.1.1 N-grams

The matching is performed at both a *bigram* and a *unigram* level.

N-grams are a pair of N words that occur next to each other in a certain sentence. N is the number specifying how many words are considered next to each other. For example, unigrams consider single words, bigrams consider two consecutive words, trigrams consider three consecutive words ...

There exists some words whose meaning is conveyed in their occurrence next to each other. Their meaning is expressed in their bigram nature.

An example is the word: *صرف صحي* *ṣrf ṣhy* which means sewage. *صرف صحي* *ṣrf ṣhy* is a bigram composed of two words, *صرف* *ṣrf* and *صحي* *ṣhy* . *صرف* *ṣrf* as a

unigram means spending. *صحي* *shy* as a unigram means healthy. In this case, taking each word on its own, will result in an incorrect result. This case highlights the importance of performing the entity and relational entity matching at both bigram and unigram levels.

At this stage, we start matching our entities and relational entities by identifying the bigrams in each tweet. We then proceed by identifying the unigrams. When a bigram match occurs, we make sure to exclude its two word components from the unigram matching, thus preventing any duplicates and matching errors. We are able to match some entities and relational entities; however, many words are yet to be tagged.

7.1.2 Morphological Analysis

To continue the task of *E* and *RE* tagging, we morphologically analyze each word to be matched. The morphological analysis allows us to determine the POS of the respective word. The POS in return allows us to infer whether the word in question serves as an entity or a relational entity; having a verb POS or a preposition POS mainly results in a relational entity tag. The rest POSs mainly result in an entity tag.

We apply morphological analysis on all the unmatched words (the words not in our *E* and *RE* corpus). After finishing the *E* and *RE* tagging, we record the results of the matching based on *N - grams* and *morphological analysis*, to discuss them thoroughly in chapter 9.

7.2 Distributional similarity

Distributional semantics is a technique for representing a word's meaning based on its linguistic contexts.

Distributional similarity gives a value for a word *w*'s meaning representation by considering the context in which *w*'s context appears. It gives the semantics (meaning) of *w*.

The *distributional representation* of a word *w* is a vector form representation for *w*. It is a recursive approach where each word predicts the other words that can appear in the same context. The word in return also predicts some other words. We can define a word by counting all the words that occur around it. *Distributional representation* of the word *w* is the representation of *w* in a vector form in N-dimensional space. Such a representation is usually called *embedding*.

7.3 Word2Vec Modal

Word2Vec is a technique which allows learning word embeddings using a two layer network. A well trained set of word vectors places all similar words close to

each other.

The Word2Vec modal is a modal we trained having its input and output as follows:

- **Input of Word2Vec:** The input is a large corpora which in our modal consists of our Yemeni collected tweets (around 40000 tweets) along with the Twitter- CBOW which is a model available in AraVec [36]. The total number of tokens used in AraVec exceeds 3,300,000,000 tokens collected from tweets, world wide web pages and Wikipedia Arabic articles. AraVec is a pre-trained distributed word representation open source project. It contains around 1,169,075,128 tokens.
- **Output of Word2Vec:** The output is the vector representation of the word.

After training the word2vec model, we are able to find any word embedding. Fig 7.1 is the word embedding for the word **عرب** *rb* , *Arab*.

By converting each word into vector format, we are able to predict the words that appear in their context. As such, we are able to predict the probability of the most appropriate word in that context.

7.4 Vocabulary Builder

The vocabulary builder is essential in building up the word2vec model. The vocabulary builder builds up its vocabulary from the corpus's raw input data by collecting all its unique words.

We used gensim library [37] to generate the word2vec for our corpus. Gensim is an open source python library for natural language processing.

7.5 Similar Words

Word2vec is able to compute the similarity between any two words present in its vocabulary. This is applied by using *model.similarity()* and passing it the two words we need to compare. *model.similarity()* will calculate the Euclidean similarity between those two words. An example is displayed in Figure 7.2, where we can notice that *Egypt* and *Tunisia* are much more similar than *Egypt* and *Apple*.

The built-in function *model.most_similar()* retrieves a set of the most similar words for a given word in the vocabulary based on their cosine similarity.

The cosine similarity is expressed in Equation 7.1. It is used to find the most similar words for a certain word. It measures the cosine of the angle Θ between two vectors A and B projected in a multi-dimensional space. The cosine similarity

```

1 # get a word vector
2 word_vector=model.wv["عرب"]
3 print(word_vector)

```

```

[-0.18730605  0.5046347  0.9388151  0.08303725  0.9475561  0.8263234
 0.36074245 -0.57606995  0.10742465 -1.1819818  0.80201125 -1.1402295
 0.49942696 -1.2690502 -0.3498572 -1.0801723 -0.4986654  0.26955378
 0.46772832 -0.53634274  0.6313674  0.7328324 -0.56021434  1.5008308
-0.2367822 -1.2323321  0.892302 -0.44083792  0.3137796 -0.92221415
 0.06105801  0.37614998 -0.9421011 -0.6656272 -0.10704076 -0.4565578
-0.85873884 -1.7335129 -1.4974713  0.5848379 -1.0641192 -0.77902246
-1.6948959 -0.50287396  1.3804884  0.3950594  0.39429492 -0.7946274
-1.481019  0.880186 -0.33428955  0.56103694 -0.97075033 -0.19615224
-0.6938657 -0.8732363 -0.8923808 -0.02900377  0.25102487  0.68465984
-0.526745 -0.91213757 -0.5456246 -0.0960575  0.4101447  0.7310291
 0.3638715 -0.89196384  1.4347217  0.908067  0.9750052 -0.6012842
-2.2266064  0.09788562 -0.10456729 -1.926564  0.82622737  1.7975557
 1.4363017 -0.32472435 -0.15363534  1.1543843  1.0443273  0.19487527
 0.2850261 -0.15302165  0.9430257  0.42737544 -0.44143477 -0.51411915
 0.73613745 -0.4908188 -0.5016154 -1.2355762 -0.627786  0.5230854
-1.2923595  0.03388799 -0.8000813  0.23631516  1.8746296 -0.72945595
 0.42834908 -1.639057  1.373044  0.8492668 -1.4409685 -0.12779494
 0.30242378 -0.6055314 -0.28583682  1.1364236 -0.5596907 -0.7616824
 0.799059  0.642374  0.82609224 -0.06875241  2.2649128 -1.635318
-0.9996487  0.41789305  0.59617734  0.04174126 -0.4981045 -0.48257643
-1.3968191 -0.77642953  0.19445828 -0.4507688 -0.7877292 -0.01780918
-0.86503935 -1.3127612 -0.05881528 -1.509829 -0.43968695 -1.0471777
 0.562624 -0.54728425  0.4847832 -0.148618 -0.07897556  1.5206399
 0.38583058  0.05891221 -1.2304696 -0.9246244 -1.8188022 -0.17599878
 0.23181073  0.7454134 -0.28241324  0.7735061  0.31242007 -0.38151827
-0.6730404 -1.4903356  0.7212757  0.98745257 -0.12138951 -0.6587063
-0.79594535 -0.33602768  0.89587057  0.18761538 -0.7490898  0.16833355
-0.6260505 -1.0174416 -0.01130947 -0.2608233  0.82551306 -1.5721133
-1.0155013 -0.82527524 -0.30447435  0.55393  0.37823188 -0.69345355
 0.32068777 -0.38632074  1.5235677  0.3427359 -0.58090466  1.5892886
 1.8572209 -0.16189197 -0.14757323 -0.4044561 -0.2173484  1.2119715
-0.5471907  0.25948164  0.2700204 -0.3609119  0.09070184 -0.11704771
 0.8662546  0.06489424]

```

Figure 7.1: Vector Representation For *عرب*

```

1 egypt = "مصر"
2 tunisia = "تونس"
3 apple = "تفاح"
4
5 print("egypt Vs. tunisia = ", model.similarity(tunisia,egypt))
6 print("egypt Vs. apple = ", model.similarity(apple,egypt))

```

```

egypt Vs. tunisia = 0.69211805
egypt Vs. apple = 0.09311087

```

Figure 7.2: Example showing the similarity between two words

will capture the angle of the word vectors provided and not their magnitude. This

similarity score ranges from 0 to 1, with 0 being the least similar and 1 being the most similar. For example, a zero degree angle between two vectors gives a similarity equal to one.

$$\text{similarity} = \cos(\Theta) = \frac{A.B}{\|A\|\|B\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}} \quad (7.1)$$

An example is displayed in Fig 7.3 where the similar words for the word تونس *twns* , Tunisia, are calculated.

```

1 # find and print the most similar terms to a word
2 most_similar = model.wv.most_similar( "تونس" )
3 for term, score in most_similar:
4 | print(term, score)
5

/usr/local/lib/python3.6/dist-packages/gensim/matutils.py:737:
  if np.issubdtype(vec.dtype, np.int):
0.7573358416557312 ليبيا
0.7550797462463379 الجزائر
0.6921180486679077 مصر
0.685177743434906 فرنسا
0.6604579091072083 موريتانيا
0.6280934810638428 السودان
0.6274716854095459 الاردن
0.6269993185997009 طرابلس
0.62644362449646 بتونس
0.6256780624389648 تركيا

```

Figure 7.3: Example for Finding Similar Words

Using the word2vec model, we are able to find all the similar words to some of our unmatched words' corpora. We record our results to be discussed in chapter 9.

7.6 Manual Inspection Iterative Approach

After applying all the above techniques to extract our entities and relational entities, we select our top frequent occurring words for evaluation. We inspect the top frequent words' tagging and manually correct them. We also inspect the tweets for possible relational or entity indicators that we have missed. We add those indicators to our corpus and rerun all the above steps to try and produce improved results.

7.7 Neural Networks

To further improve our results, we apply a neural network approach. We employ the use of Long Short-Term Memory (LSTM) networks. LSTM is a type of recurrent neural network (RNN). It is characterized by being able to learn the ordered sequence of its input.

7.7.1 RNNs

A basic neural network's inputs and outputs are independent of each other. However, there exists cases which require remembering the previous word to be able to predict the next word in a sequence. Such cases, require the use of RNNs.

RNNs are characterized by having a *hidden layer*, figure 7.4. The hidden state makes it possible for the network to remember the sequence's information. Recurrent neural networks are able to work with sequential data. In an RNN, the output resulting from the previous step is the input to the current step as displayed in figure 7.5 [1], resulting in a third output. This expresses the recurrency.

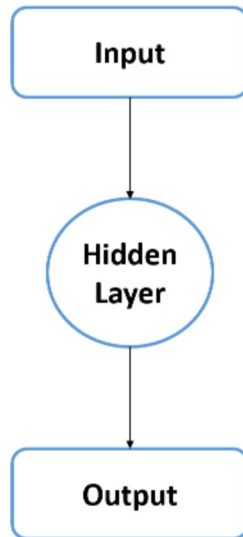


Figure 7.4: Hidden State

7.7.2 LSTM

In an LSTM network, each hidden layer is replaced by an LSTM cell. The LSTM takes two inputs and produces two outputs. The first input is the *word embeddings*, the second input is the hidden state and cell state. Initially, the hidden states and cell states are random values. The outputs are the hidden state and the cell state. Those outputs along with the next input word are fed

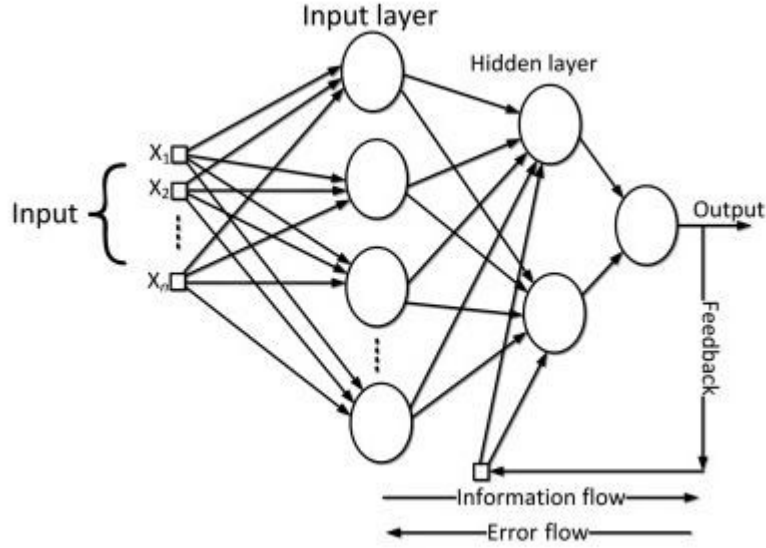


Figure 7.5: RNN Diagram Obtained from [1]

into the second time step of the LSTM. In return, this will produce two vectors which will be fed along with the next input word to the third time step of the LSTM ...

The input of our LSTM modal are our tweets split into word tokens. We obtain the word embedding for each of the words from Mazajak Embeddings [38], trained on 100 million Arabic tweets, to extract a feature vector of size 300. We feed the obtained embedding into an LSTM.

We split the tagged labeled words into a training data set and a testing data set. The training data set represents 80% of our labelled words data set. The testing data set represents 20% of our labelled words data set.

The output of the LSTM will classify each word token into either an entity or a relational entity. This LSTM is a birectional LSTM (from left to right and from right to left). The LSTM has a hidden size of 500.

We record our results to be discussed in section 9.

Chapter 8

Leveraging Partial Tuples

In this chapter, we discuss how to further improve the extraction of entities and relational entities. We work on leveraging partial tuples to enhance our information extraction.

To further decrease the number of unmatched words and to improve our results, we compute and add the following partial tuples to our matching:

- tuple $(e1, e2, ?)$
- tuple $(e1, ?, r)$

We discuss how we leverage each of these tuples in the following sections:

8.1 Partial Tuple $(e1, e2, ?)$

We leverage the partial tuple $(e1, e2, ?)$ to try and enhance our methodology and to further improve our results. The partial tuple $(e1, e2, ?)$ is a tuple composed of $e1$ (*entity1*), $e2$ (*entity2*), and an *unknown*. We explain the steps applied in this technique by illustrating its respective flow diagram. This method's flow diagram is illustrated in Figure 8.1. Each step is also explained in the list below.

- We start by computing the distributional similarity index D , as discussed in section 7.2, for all the tweets.
- We then compute E and R , the entities and relational entities in the tweets respectively.
- We check for each partial tuple $(e1, e2, ?)$ in tweet t in the relations R .
- We compute $w1$, the word in tweet t with minimum distributional similarity D distance to $e1$.
- We compute $w2$, the word in tweet t with minimum D distance to $e2$.

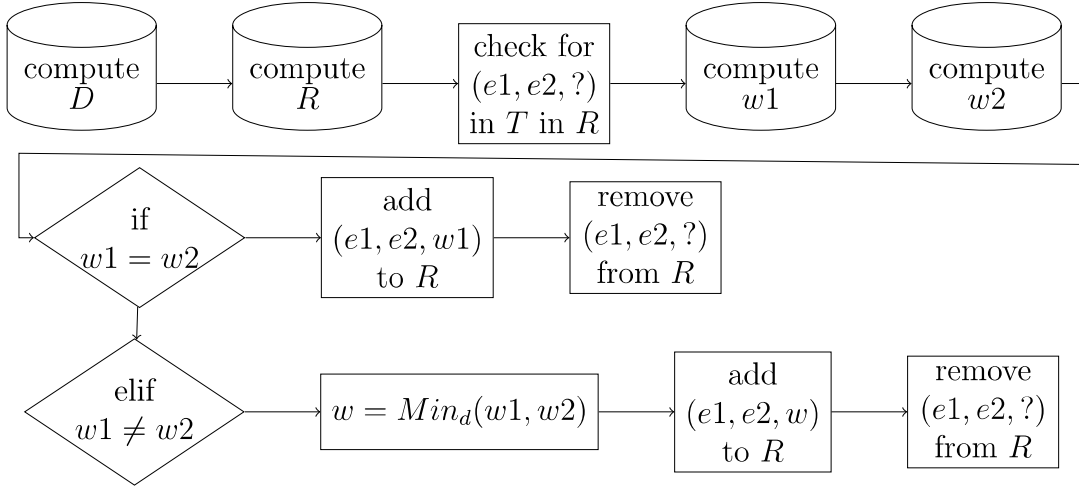


Figure 8.1: Adding partial tuples: $(e1, e2, ?)$

- If $w1$ is equal to $w2$, we add $(e1, e2, w1)$ to the R set and we remove the partial tuple containing an unknown, $(e1, e2, ?)$, from the R set.
- Else (*if $w1$ is not equal to $w2$*), we pick w to be either $w1$ or $w2$. We choose w to be the word with the minimum distance between $w1$ and $w2$. We then add the tuple $(e1, e2, w)$ to R and remove the partial tuple containing an unknown, $(e1, e2, ?)$, for all the tweets, from the R set.

We record our results to be discussed in section 9.

8.2 Partial Tuple $(e1, ?, r)$

We then leverage the partial tuple $(e1, ?, r)$ to try and enhance our methodology and to further improve our results. The partial tuple $(e1, ?, r)$ is a tuple composed of $e1$ (*entity1*), an *unknown*, and r (*relation*). We explain the steps applied in this technique by illustrating its respective flow diagram as well. This method's flow diagram is illustrated in Figure 8.2. Each step is also explained in the list below. The following method is illustrated in Figure 8.2.

- We start by computing the distributional similarity index D , discussed in section 7.2, for all the tweets.
- We compute R , the entities and relational entities in the tweets.
- We check for each partial tuple $(e1, ?, r)$ in tweet t in R .
- We compute $w1$, the word in tweet t with the minimum D distance to $e1$.
- We compute $w2$, the word in tweet t with the minimum D distance to r .

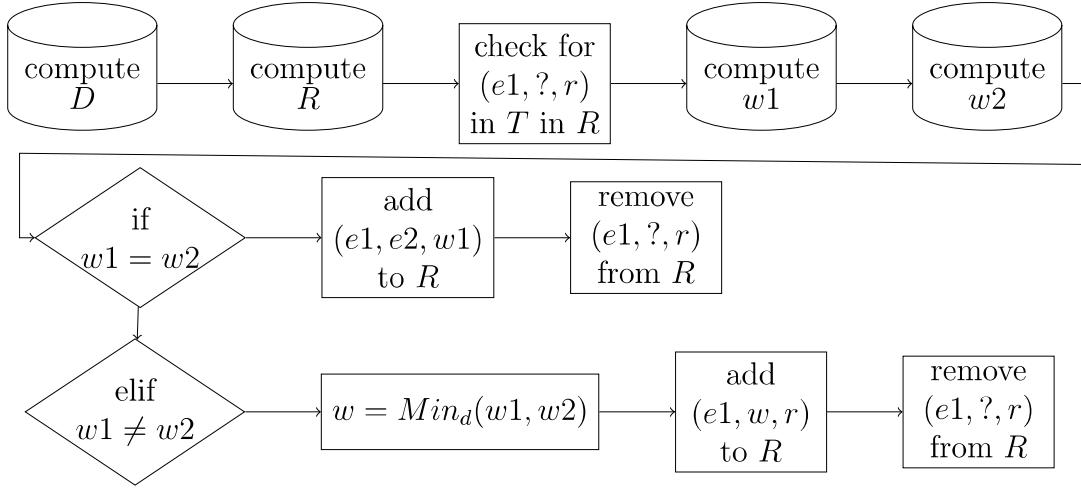


Figure 8.2: Adding partial tuples: $(e1, ?, r)$

- If $w1$ is equal to $w2$, we add $(e1, w1, r)$ to the R set and we remove the partial tuple containing an unknown $(e1, ?, r)$.
- Else (*if $w1$ is not equal to $w2$*), we pick w to be either $w1$ or $w2$. We choose w to be the word with the minimum distance between $w1$ and $w2$. We then add the tuple $(e1, w, r)$ to the R set and remove the partial tuple containing an unknown $(e1, ?, r)$, for all the tweets, from the R set.

We record our results to be discussed in section 9.

Chapter 9

Results

In this chapter, we discuss and analyze the results of all our applied methodology techniques on different evaluation metrics.

9.1 Testing Data Set

We randomly selected 50 random tweets and manually annotated them to either *entity* or *relational entity*. This annotated set will serve as our testing data set.

9.2 Results Evaluation

9.2.1 Confusion Matrix

A confusion matrix is a performance measurement for classification where the output can be two or more classes. The classes in our case are: entity and relational entity. A confusion matrix is represented by a table displayed in figure 9.1 with 4 different combinations of predicted and actual values. A *true positive* is when the model correctly predicts the positive class, *E*. A *true negative* is when the model correctly predicts the negative class, *RE*.

A *false positive* is when the model incorrectly predicts the positive class. A *false negative* is when the model incorrectly predicts the negative class.

9.2.2 Evaluation Metrics

To evaluate our results, we measure the *recall* and *precision* of our model's tags with respect to our testing data set tags.

Recall, expressed in equation 9.2, measures the percentage of total relevant results correctly classified by the algorithm. It measures the number of correct predictions out of all the predictions made. A low recall indicates many false negatives.

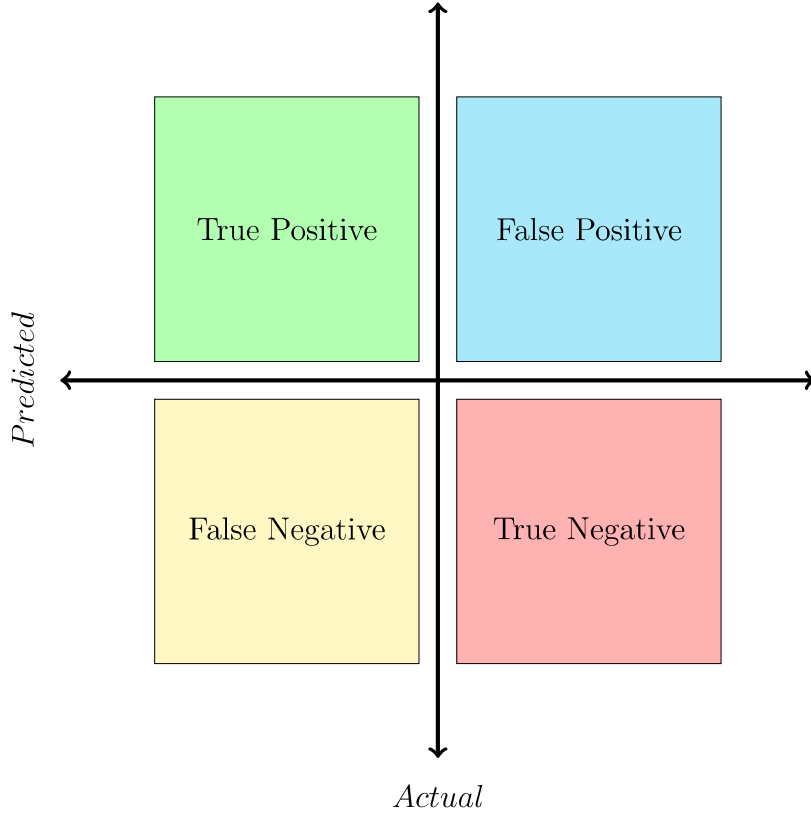


Figure 9.1: Confusion Matrix

Precision, expressed in equation 9.1, measures the percentage of results which are relevant. A low precision indicates a large number of false positives. Maximizing both of these metrics at the same time is difficult, as one comes at the cost of another. For simplicity, we use the *F-1 score* metric, expressed in equation 9.3. The F-1 score is a harmonic mean of the precision and recall. It measures the balance between the precision and the recall. An F-1 score has a best value of 1 and a worst value of 0. A low F1 score is an indication of both poor precision and poor recall.

$$Precision = \frac{True\ Positive}{Actual\ Results} \quad or \quad \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (9.1)$$

$$Recall = \frac{True\ Positive}{Predicted\ Results} \quad or \quad \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (9.2)$$

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (9.3)$$

We also record the number of words, we were unable to match as *E* or *RE*.

9.3 Initial data

In table 9.1, we record the initial number of tweets we started with: around 40000 tweets. We also record having around 50000 unique words.

Total Number of Tweets	37001
Total Number of Unique Words	54458

Table 9.1 Initial Data

9.4 Evaluation Results

For each stage, we measure our results on:

- recall
- precision
- Fscore
- number of unmatched words

Stages:

- Stage 1: after tagging our tweets by matching them with our E and RE data set, as discussed in section 7.1.2 and section 7.1.1.
- Stage 2: after applying distributional similarity then matching with our E and RE data set, as discussed in section 7.2.
- Stage 3: after manually inspecting our top frequent words and retraining as discussed in section 7.6 then applying distributional similarity.
- Stage 4: after applying neural networks on our data, as discussed in section 7.7.
- Stage 5: after leveraging partial tuples to further extract entities and relational entities, as discussed in chapter 8.

The measured results are recorded in table 9.2. we notice a strong increase in the precision and recall at every stage of our methodology.

	First Matching			
	Before DS		After DS	
	E	RE	E	RE
Recall	59	56	62	58
Precision	87	54	88	55
Fscore	70	55	73	56
Unmatched Words >20	2940		594	

(a) Results After First Matching

	After Evaluation			
	Before DS		After DS	
	E	RE	E	RE
Recall	80	67	80	79
Precision	91	74	91	74
Fscore	85	70	85	76
Unmatched Words >20	562		480	

(b) Results After Manual Inspection

	After Applying Neural Networks	
	E	RE
Recall	82	83
Precision	89	85
Fscore	85	84
Unmatched Words >20	0	

(c) Results After Applying Neural Networks

	After Leveraging Partial Tuples	
	E	RE
Recall	84	85
Precision	91	87
Fscore	87	86
Unmatched Words >20	434	

(d) Results After Leveraging Partial Tuples

Table 9.2 Evaluation Metrics Results

9.5 Numeric Evaluation Results

9.5.1 Recall

In the case of entity tagging, we initially had a *recall* of 59 at stage 1. The *recall* increased to 62, 80, 82, 84 at stages 2, 3, 4, and 5 respectively displayed in tables 9.2a, 9.2b, 9.2c, 9.2d. The sequence of numbers is increasing with the increase of every stage, indicating the efficiency of our methodology.

In the case of relational entity tagging, we initially had a *recall* of 56 at stage 1. The *recall* increased to 58, 79, 83, 85 at stages 2, 3, 4, and 5 respectively displayed in tables 9.2a, 9.2b, 9.2c, 9.2d. The sequence of numbers is increasing with the increase of every stage, indicating the efficiency of our methodology.

9.5.2 Precision

In the case of entity tagging, we initially had a *precision* of 87 at stage 1. The *precision* increased to 88, 91, 89, 91 at stages 2, 3, 4, and 5 respectively displayed in tables 9.2a, 9.2b, 9.2c, 9.2d. The sequence of numbers is increasing with the increase of every stage, indicating the efficiency of our methodology.

In the case of relational entity tagging, we initially had a *precision* of 54 at stage 1. The *precision* increased to 55, 74, 85, 87 at stages 2, 3, 4, and 5 respectively displayed in tables 9.2a, 9.2b, 9.2c, 9.2d. The sequence of numbers is increasing with the increase of every stage, indicating the efficiency of our methodology.

9.5.3 Fscore

In the case of entity tagging, we initially had a *Fscore* of 70 at stage 1. The *Fscore* increased to 73, 85, 85, 87 at stages 2, 3, 4, and 5 respectively displayed in tables 9.2a, 9.2b, 9.2c, 9.2d. The sequence of numbers is increasing with the increase of every stage, indicating the efficiency of our methodology.

In the case of relational entity tagging, we initially had a *Fscore* of 55 at stage 1. The *Fscore* increased to 56, 76, 84, 86 at stages 2, 3, 4, and 5 respectively displayed in tables 9.2a, 9.2b, 9.2c, 9.2d. The sequence of numbers is increasing with the increase of every stage, indicating the efficiency of our methodology.

The *Fscore* is a harmonic measure of the *precision* and *recall*. Obtaining an increasing *Fscore* at each stage is crucial and important.

9.6 Evaluation Results Plot

9.6.1 Recall, Precision and F1score Plot

To better visualize our results and to facilitate their analysis, we plot our results and display them in figure 9.2.

We can notice how our $F1 - score$, marked with a grey color, is increasing with every stage of our methodology. The increase in the $F1 - score$ indicates that our approach is successful.

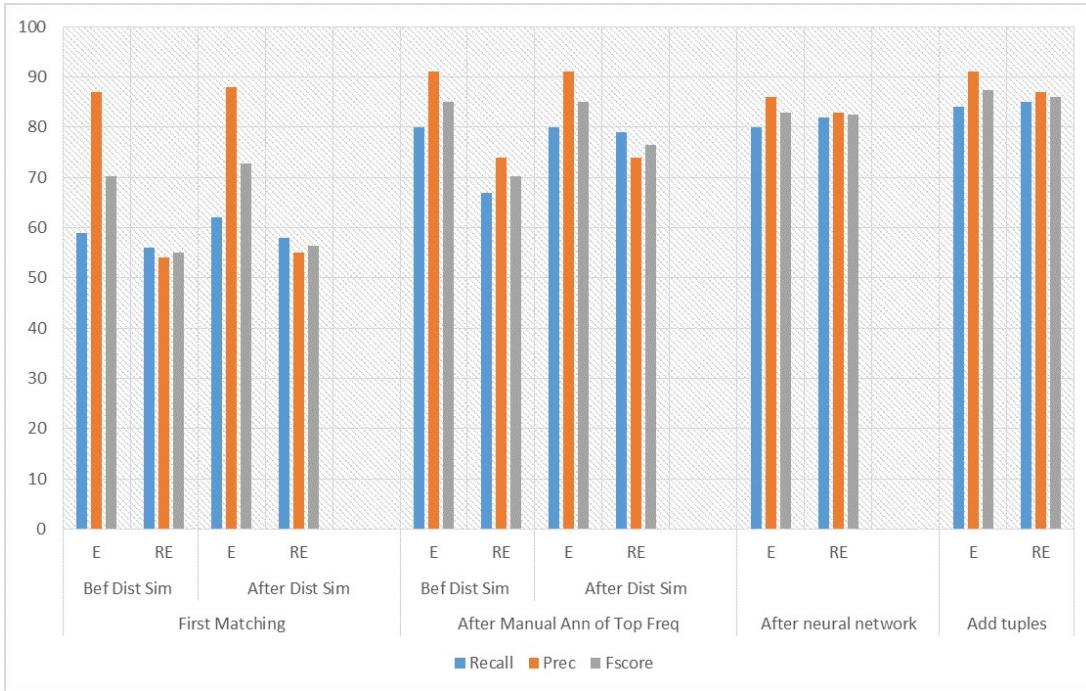


Figure 9.2: Results plot showing the recall, precision and fscore for the E and RE extraction at each stage in our applied techniques

9.6.2 Unmatched Words Plot

We care to make certain that the number of our unmatched words is decreasing at each stage of our methodology as well. As such, we plot the number of unmatched words at each stage and display them in a bar graph in figure 9.3. We can record the very obvious decrease in the number of unmatched words at every stage of our methodology which is very promising.

It is worth mentioning here that we started with 54458 unique words in our cleaned tweet data set. At the end of our methodology, the number of unmatched words recorded is 434 words. Thus, we were able to match 99.2% of our initial words with a Fscore of 87% for E and 86% for RE .

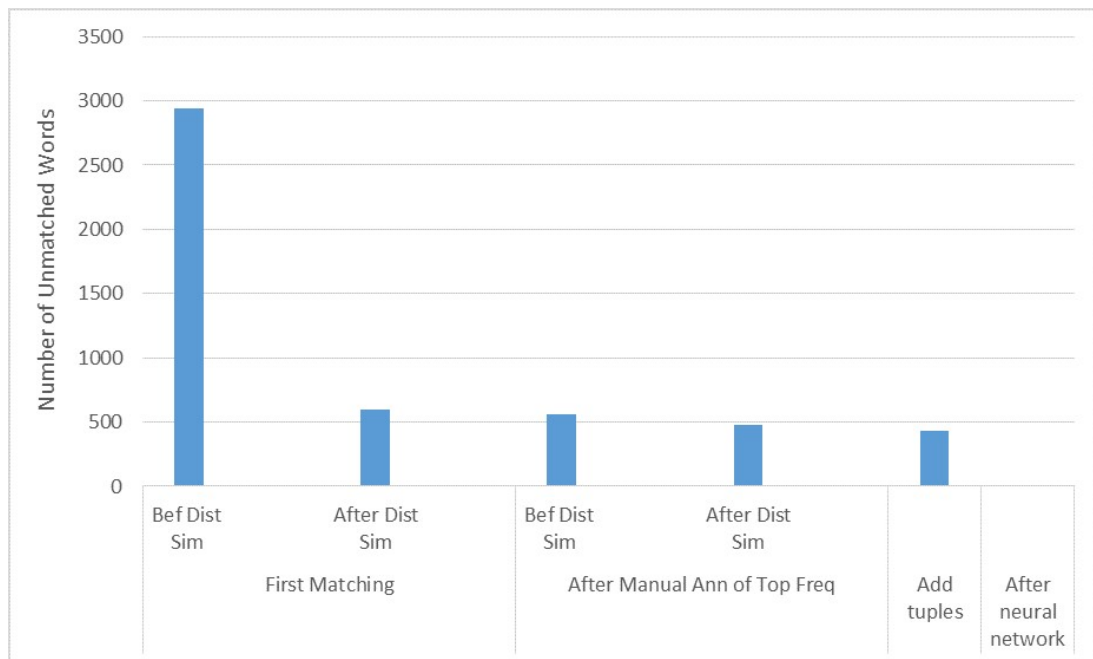


Figure 9.3: Unmatched Words Plot showing the number of unmatched words at each stage in our applied techniques

Chapter 10

Graph Analysis and Insights

Graph visualization is a means of providing knowledge about the data that resulted from the information extraction. After being able to successfully perform information extraction on our tweets, we have the basis needed to build their graph representation.

In this chapter, we visualize some of our tweets by drawing their respective graphs. We apply some graph algorithms to better illustrate the graph. We then analyze and draw insights from the constructed graph.

10.1 Graph Data

As discussed in section 4.3, a graph is composed of nodes linked together by edges. The nodes of our graph are our extracted entities. Our data set contains around 40,000 tweets with around 10,000 entities. For simplicity, we construct a graph representing some of our tweets. We select the top frequent occurring word entities in our tweet data to set as the nodes of our graph.

The graph in figure 10.1 displays the top frequent word entities in our data. The plot shows the number of occurrences (frequency) of our most occurring words.

The edges of our graph are our extracted relational entities. We will construct the graph visualizing the words in figure 10.1. The graph is a directed graph linking the entities by the extracted relational entities.

Table 10.1a contains the top frequent entities which represent and will construct our nodes. Table 10.1b contains the edges which link our nodes together.

The graph $G = (E, \mathcal{R})$ is a graph where:

- E represents all the entities
- \mathcal{R} represent the relational entities in the tweet text that match R
- \mathcal{R} specifies the edges of G

Id	Label
1	مؤامرة <i>m>wāmrah</i>
2	رئيس <i>r>yys</i>
3	اخونج <i>āḥwnġ</i>
5	جيش <i>ġyš</i>
17	ثاني <i>tāny</i>
20	دنبوع <i>dnbw^c</i>
22	منصور هادي <i>mnšwr hādy</i>
23	علي صالح <i>ly ṣālḥ</i>
35	صرف صحي <i>ṣrf ṣḥy</i>
37	فساد <i>fsād</i>
39	طرق <i>tṛq</i>
40	يمن <i>ymn</i>
41	سيل <i>syl</i>
42	عصابة <i>ṣābh</i>
46	فقر <i>fqr</i>
47	اول <i>āwl</i>
49	مجرم <i>mġrm</i>
50	صنعا <i>ṣn^cā</i>
51	حوثي <i>ḥwtly</i>
52	تدمير <i>tdmyr</i>
53	عفاش <i>fāš</i>
55	حقيقة <i>ḥqyqh</i>
56	حكومة <i>ḥkwmh</i>
57	واسطة <i>wāsth</i>
58	شمال <i>šmāl</i>
59	روافض <i>rwāfḍ</i>
60	ارهاب <i>ārḥāb</i>
61	دمار <i>dmār</i>
62	سلاح <i>slāḥ</i>
63	سرقة <i>srqh</i>

(a) nodes table

Source	Target	Label
20	2	
53	5	قاد <i>qād</i>
53	49	
53	52	سعى الى <i>s^cā ālā</i>
52	40	
53	55	زيف <i>zyf</i>
20	3	صنع <i>ṣn^c</i>
37	56	
37	20	
37	3	
56	53	
53	37	
56	37	
37	57	
37	46	
3	60	
3	42	
20	51	ادخل <i>ādḥl</i>
51	40	
51	59	
58	51	ايد <i>āyd</i>
37	41	
37	35	
41	39	
53	41	
53	61	
51	61	
51	62	اخذ <i>āḥḍ</i>
53	37	
37	63	
53	37	
53	37	
53	1	
53	23	القب <i>lqb</i>
20	2	
1	2	
53	2	
41	35	
20	17	
17	2	
20	22	القب <i>lqb</i>

(b) edges table

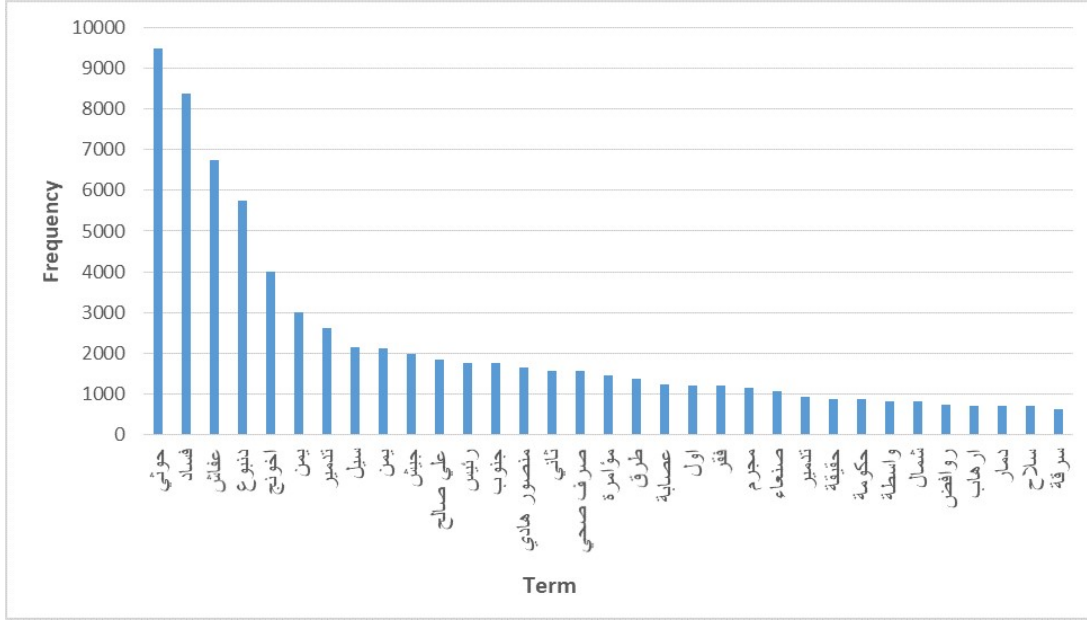


Figure 10.1: Selected Top Frequent Words

- $\mathcal{R} \subseteq V \times V \times L$ where:
 - L is a set of contextual labels

10.2 Graph Construction

We set our graph by defining our nodes and edges, using graphviz package [39]. Graphviz is an open source graph visualization software. For a clear graphical interface, we illustrate our graph using Gephi [40]. Gephi is an open source software for visualizing and manipulating graphs. The data in table 10.1, produces the graphs illustrated in figure 10.2.

In a large scale graph, we can have a big number of nodes and edges. Illustrating a graph without any processing and manipulation is futile. In the graph, in figure 10.2, we observe many connected nodes, but we are unable to infer any other information. The nodes are condensed and overlapping each other.

10.3 Graph Visualization Algorithms

To understand the graph and to be able to infer and draw relevant insights we apply the following algorithms:

- Force atlas
- Graph rank degree

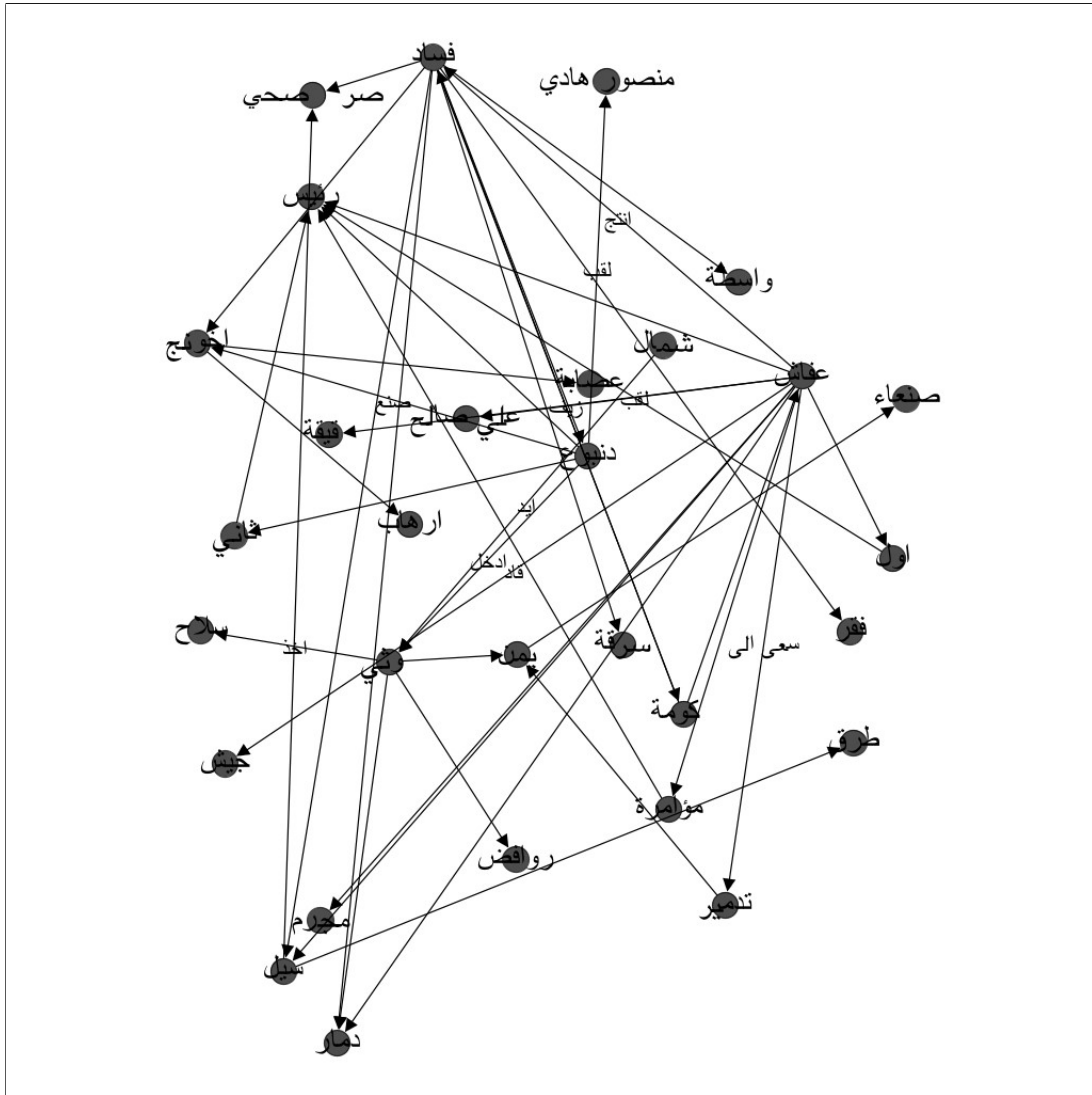


Figure 10.2: Graph Illustration

- Average path length and betweenness centrality
- Community Detection

We discuss each algorithm and show its output and benefit to our graph in the following subsections.

10.3.1 Force Atlas Algorithm

The *leaves* of a graph are the nodes with no children. The nodes surrounding the leaves are usually one of the sources of visual clutter. Observing figure 10.2, we confirm visual clutter emphasis around the leaves.

We apply the *force atlas* algorithm [41] to better visualize our graph. *Force atlas* attracts the linked nodes next to each other and pushes apart the non-linked nodes. It positions poorly connected nodes next to very connected nodes. This produces better readability of the graph. Figure 10.4 shows the graph after applying *force atlas*. Force atlas calculates the degree of the nodes (the count of connected edges) in the repulsion. This reduces the clutter surrounding the leaves. The formula of the repulsion force (F_r) between two nodes n_1 and n_2 is expressed in equation 10.1, where K_r is a defined coefficient, $deg(n)$ is the degree of a node n , and $d(n_1, n_2)$ is the distance between two nodes n_1 and n_2 .

$$F_r(n_1, n_2) = k_r \frac{(deg(n_1) + 1)(deg(n_2) + 1)}{d(n_1, n_2)} \quad (10.1)$$

In figure 10.3, we observe the graph generated after applying force atlas algorithm with a repulsion strength of 10,000. We can observe that the nodes surrounding the leaves are now more obvious. The linked nodes are next to each other the non-linked are a bit apart.

By visual observation, we try different repulsion strengths on the cluttered nodes until we are able to reach a well readable graph.

We increase the repulsion strength and set it to 20,000 for better readability and display it in figure 10.4. The graph is much more readable and less cluttered now.

10.3.2 Graph Rank Degree

Another important indicator to draw insights in graph theory is to determine the importance of a node relative to other nodes of the graph. One of way of achieving this is by ranking the nodes according to their degree of connectivity. In an undirected graph, the degree of a node n is the number of edges the node n has. It is the sum of edges for the node n . Our graph is a directed graph, where in this case, each node has an *in-degree* and an *out-degree*. The *in-degree* is the number of edges coming to a node n . The *out-degree* is the number of outgoing edges from a node n . The total degree of the node n is the sum of its *in-degree* and *out-degree*.

We find the degree for each of the nodes in our graph. A relation (edge) is connecting two entities(nodes), so if an entity has a high degree, it means it has many entities as neighbours. Table 10.2, lists each entity node in our graph. It displays the in-degree, out-degree, and final degree for each respective node.

We apply the rank degree algorithm [42] to rank the most connected nodes in our graph. The algorithm, colors each node according to its degree. The darkness in the color of a node is an indicator of the high rank of that respective node.

Applying the rank degree algorithm, produces the graph displayed in figure 10.5.

Observing the graph 10.5 and the degrees for each node in table 10.2, we are able to infer that:

Node	In-Degree	Out-Degree	Degree
عفّاش <i>fāš</i>	1	11	12
فساد <i>fsād</i>	2	8	10
دنبوع <i>dnbw</i>	1	5	6
حوثي <i>hwṭy</i>	2	4	6
رئيس <i>ryys</i>	5	0	5
اخونج <i>āhwnǵ</i>	2	2	4
سيل <i>syl</i>	2	2	4
يمن <i>ymn</i>	2	1	3
حكومة <i>ḥkwmh</i>	1	2	3
مؤامرة <i>ṃwāṃrh</i>	1	1	2
ثاني <i>tāny</i>	1	1	2
صرف صعي <i>šrf ṣ̌ḥy</i>	2	0	2
اول <i>āwl</i>	1	1	2
تدمير <i>tdmyr</i>	1	1	2
دمار <i>dmār</i>	2	0	2
جيش <i>ǵyš</i>	1	0	1
منصور هادي <i>mnšwr hādy</i>	1	0	1
علي صالح <i>ly ṣālḥ</i>	1	0	1
طرق <i>trq</i>	1	0	1
عصابة <i>ṣābh</i>	1	0	1
فقر <i>fqr</i>	1	0	1
مجرم <i>mǵrm</i>	1	0	1
صنعا <i>ṣnā</i>	1	0	1
حقيقة <i>ḥqyqh</i>	1	0	1
واسطة <i>wāsth</i>	1	0	1
شمال <i>šmāl</i>	0	1	1
روافض <i>rwāfd</i>	1	0	1
ارهاب <i>ārḥāb</i>	1	0	1
سلاح <i>slāḥ</i>	1	0	1
سرقة <i>srqh</i>	1	0	1

Table 10.2 Degree of Each Entity Node

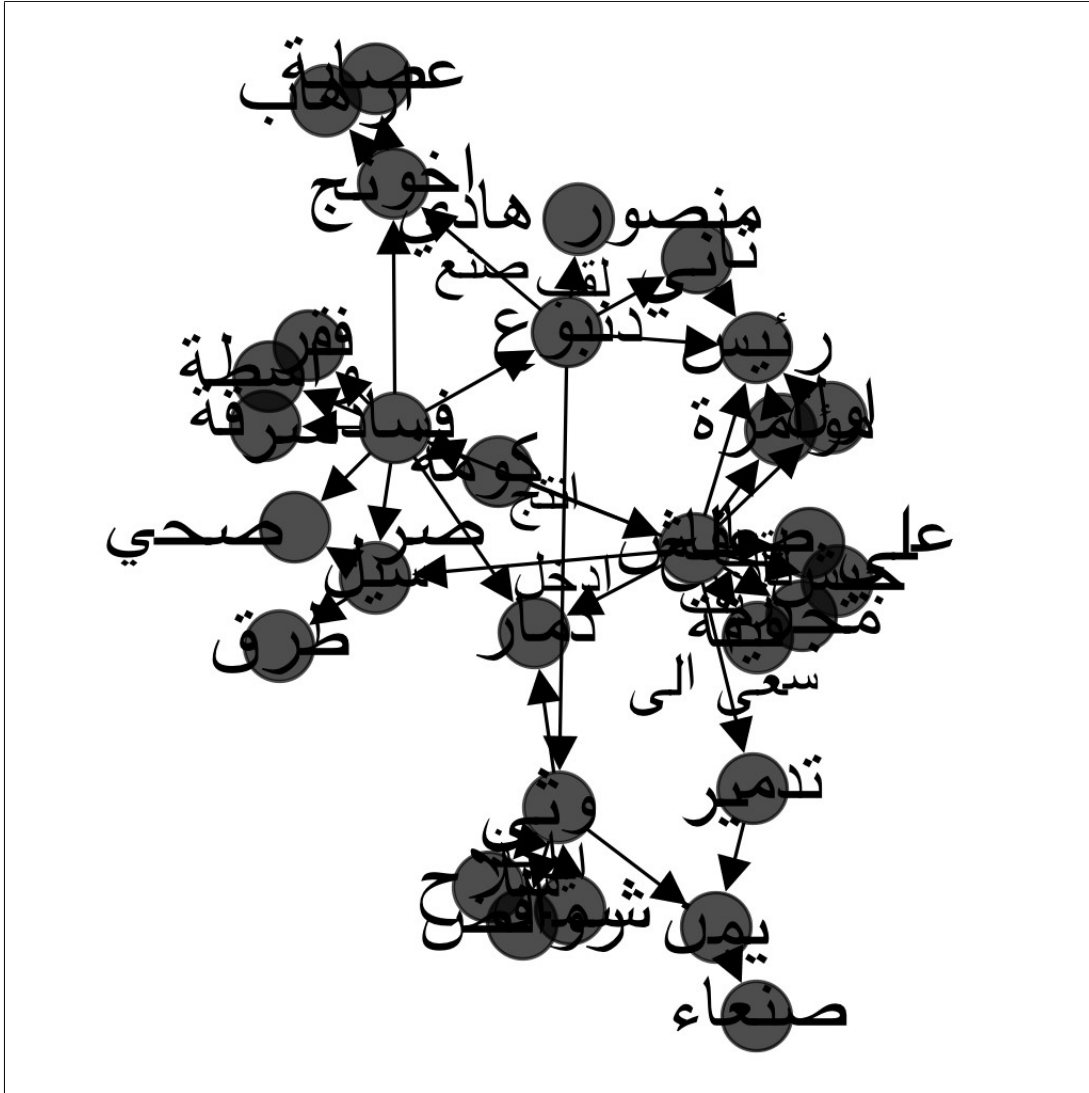


Figure 10.3: Graph Illustration After Force Atlas Algorithm with a Repulsion Strength of 10,000

- Entity عفاش *fāš* , (*Afash*), and entity فساد *fsād* , (*corruption*), have the highest degrees.
- Entity دنبوع *dnbw* , حوثي *hwty* , (*Houthi*), entity رئيس *ryys* , (*president*),

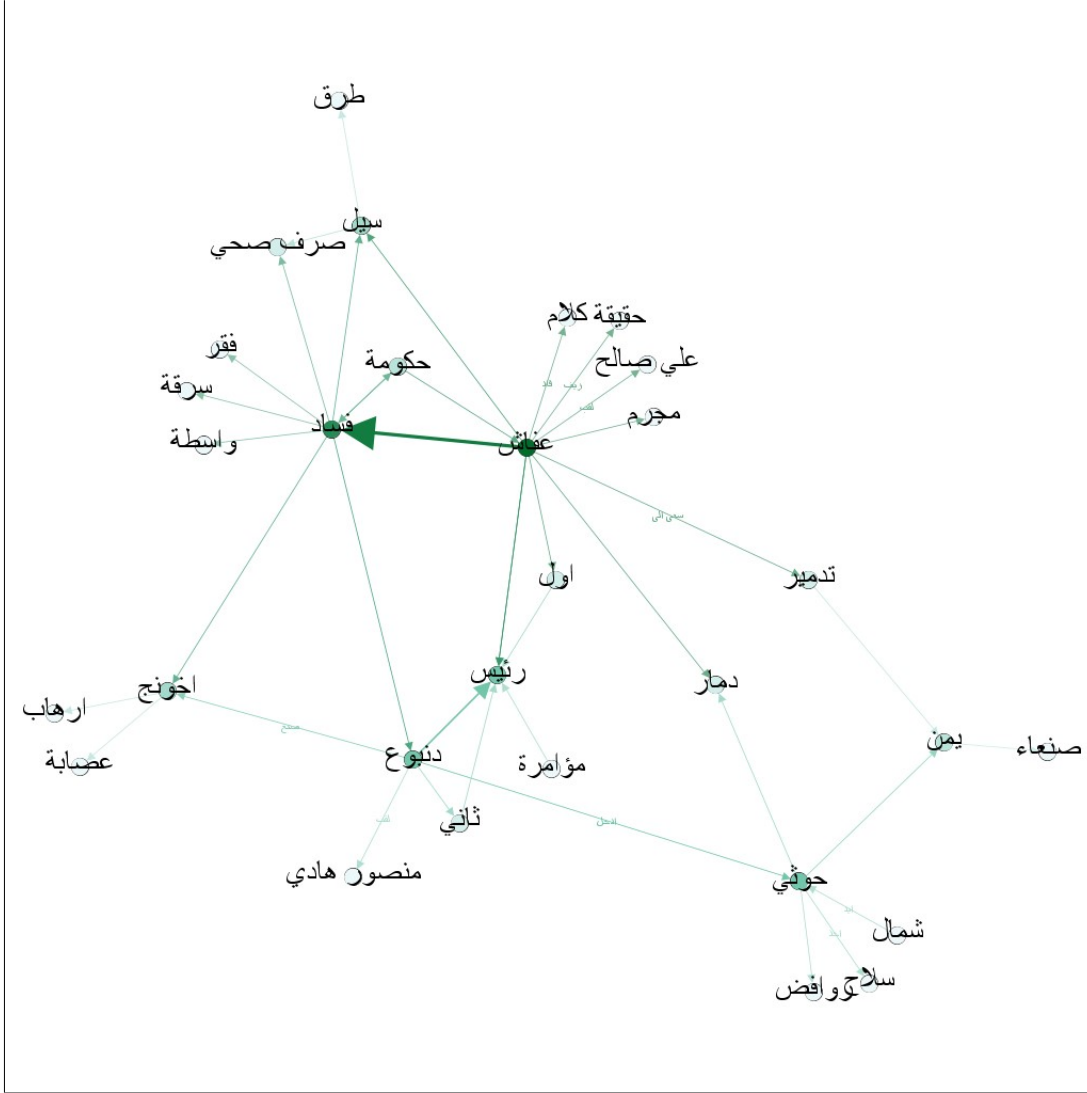


Figure 10.5: graph illustration after rank degree

In a graph, the length of a path is the number of edges the path contains.

The average path length, l_{av} , is the average of the shortest path length, averaged over all pairs of nodes. The average path length is expressed in equation 10.2, where N is the total number of nodes in the graph, $d_{n_1 n_2}$ is the shortest path length between node n_1 and node n_2 , and the summation is over all pairs of distinct nodes.

$$l = \frac{1}{N(N-1)} \sum_{n_1 \neq n_2} d_{n_1 n_2} \quad (10.2)$$

Betweenness Centrality:

The concept of betweenness centrality [43] works by:

- taking all the shortest paths between all nodes in the graph

- in each path, if a node n_1 is traveled, adding for the node n_1 one point
- once all the paths are covered, a ranking is achieved, showing which nodes are traveled a lot and which are not

This explains the concept of betweenness centrality. A node having a high rank number, has a high betweenness centrality.

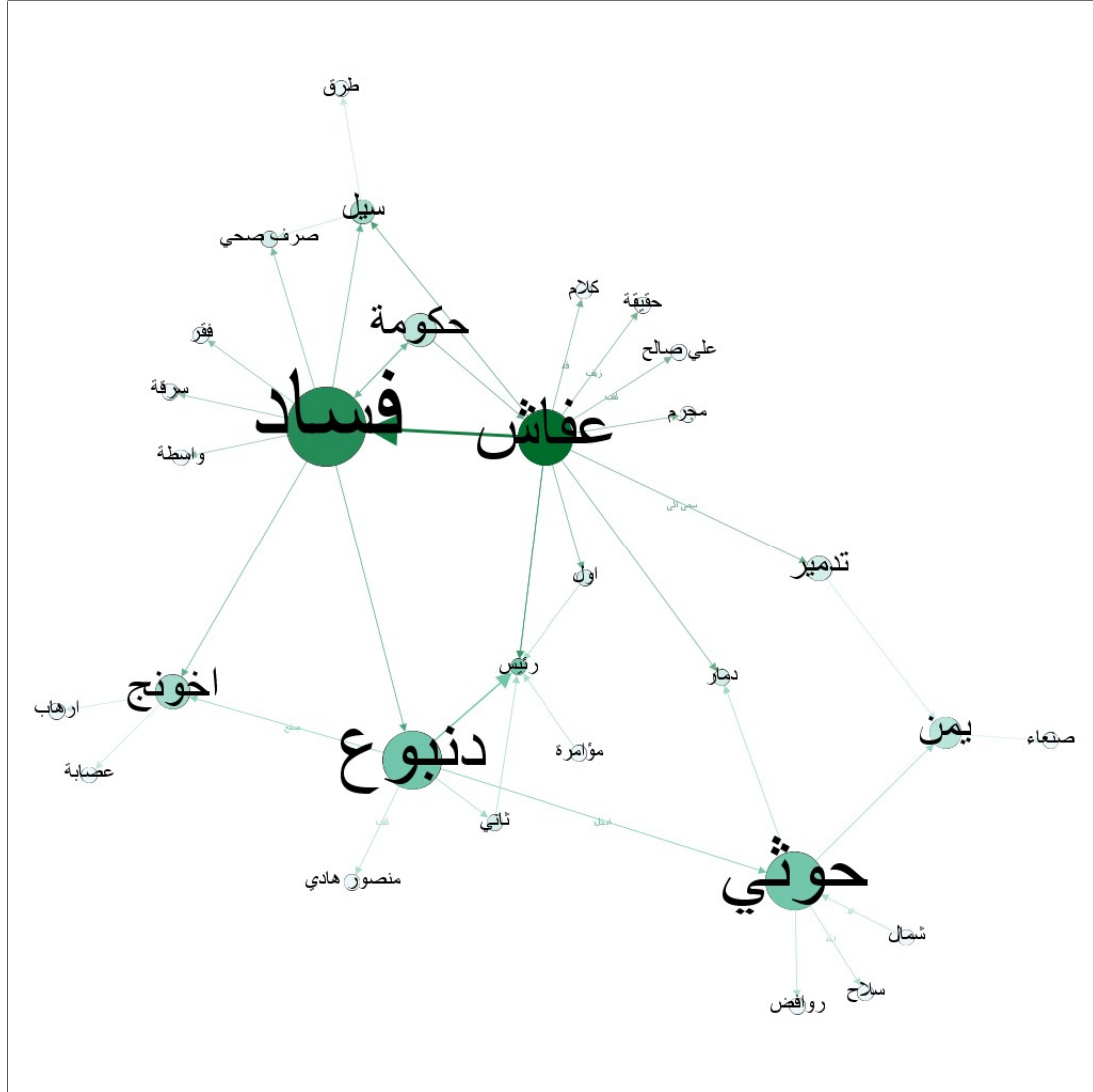


Figure 10.6: Graph Illustration After Ranking the Node Size on Betweenness Centrality

Betweenness centrality represents the degree of which nodes stand between each other. A node with high betweenness centrality has more control and influence over the graph, because more information passes through it to the other nodes.

We now rank our nodes by size according to their betweenness centrality.

Figure 10.6, displays the graph after ranking the nodes on the betweenness centrality metric. The larger the node, the greater is its betweenness centrality, thus the greater is its influence.

Observing the graph, we notice that the nodes: *فساد* *fsād* , *عفاش* *fāš* , *دنبوع* *dnbw* , and *حوثي* *hwti* have the greatest size. This means those nodes have the highest betweenness and the highest influence.

10.3.4 Community Detection

The graph generated up this stage, is much more clear than the initial graph we started with.

The final graph algorithm we are applying, is community detection. A community can be defined as the following: "A community, with respect to graphs, can be defined as a subset of nodes that are densely connected to each other and loosely connected to the nodes in the other communities in the same graph." [44].

Community detection is also known as clustering. It groups nodes into communities (also called clusters). The nodes inside the community or cluster have many connections and the nodes between the marked communities have few connections.

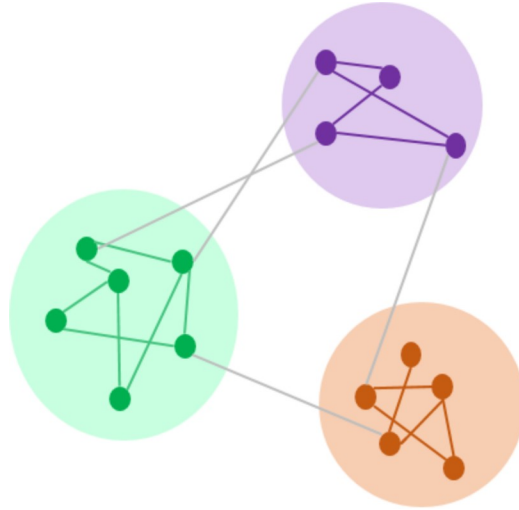


Figure 10.7: Example of Graph Detecting and Highlighting of Communities

In figure 10.7, we highlight the communities of a random graph to better illustrate the concept of community detection. The nodes of this graph are partitioned into three communities, each labeled with a different color. The majority of the edges are inside the detected communities. The minority of the edges are between the detected communities.

We want to colorize the communities of our graph to somehow resemble the community structure in the graph in figure 10.7. Community detection is achieved by applying the Louvain method [45]. The Louvain method maximizes the modularity score for each community. It evaluates how dense the connected nodes within a community are.

Figure 10.8b, displays the graph after detecting the communities and colorizing them. Each community is identified by a randomly generated color displayed in figure 10.8a.

By observing the resulting graph in figure 10.8, we notice four major communities in our graph. We can now analyze each community separately to infer the topic it discussed. We can then combine our findings and understand the information, the graph as a whole, is providing.

We can also represent some of the nodes with their respective images by deploying the `google_images_download` Python script [46]. The graph in figure 10.9 displays the final graph with the main nodes represented by their respective images.

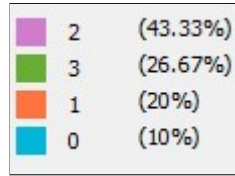
10.4 Graph Insights

After observing the graphs in figure 10.8 and figure 10.9, we can infer some interesting insights.

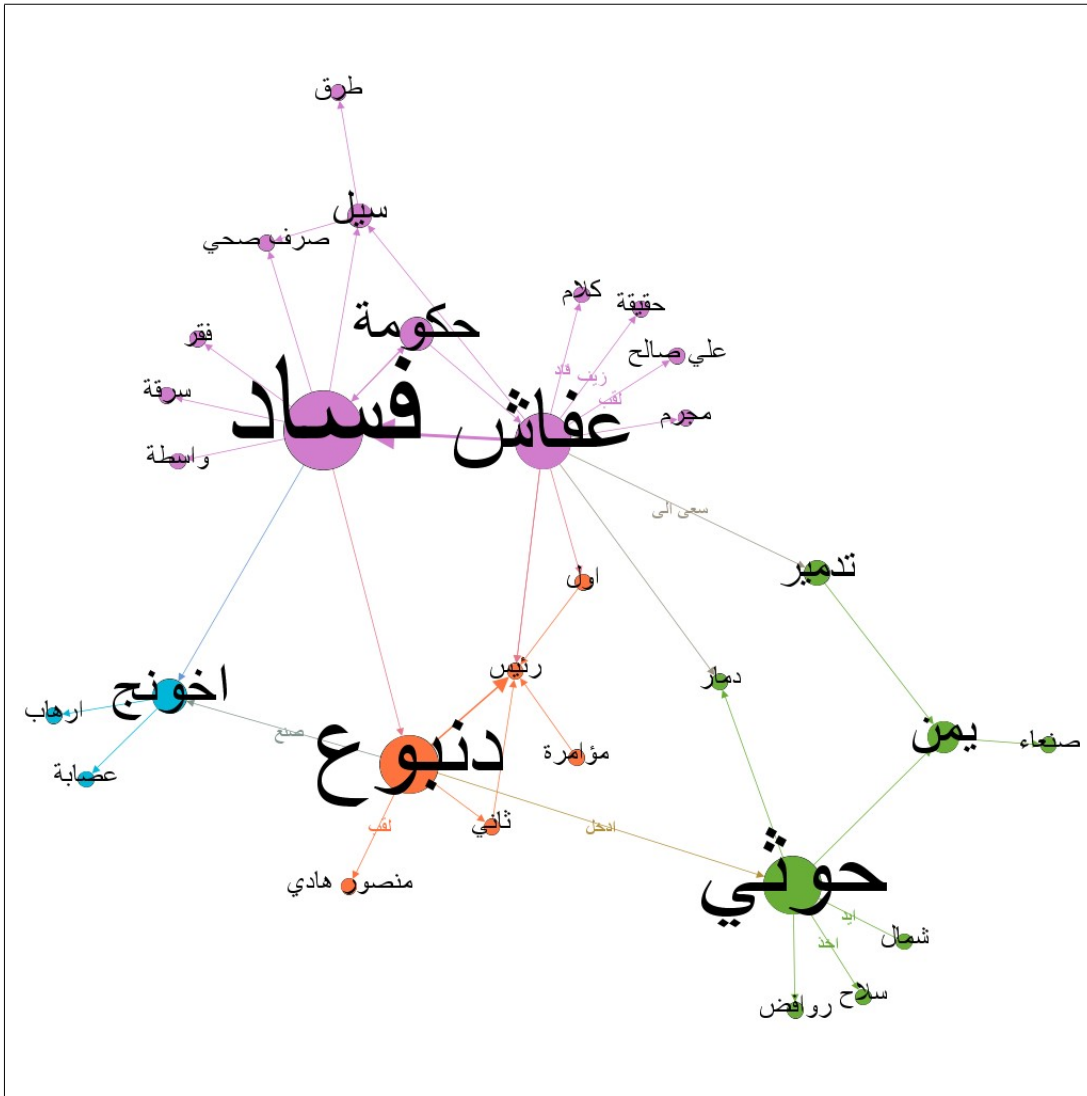
It is important to note that the insights represent the data collected from the tweets, and thus do not necessarily reflect our opinion to the topics inferred and discussed.

We gather that **عفّاش** *fāš* is the nickname of the former president of Yemen **علي صالح** *ʿalī ṣāliḥ*. **عفّاش** *fāš* was the first president of Yemen. He was heavily associated with corruption according to the people’s opinion. His government is directly linked to the corruption in Yemen which is expressed in poverty, thievery, criminal activity, flooding of roads and sewer system, and influential power exerted to force authority.

دنبوع *dnbwʿ* is also linked to all the corruption forms we mentioned before. **دنبوع** *dnbwʿ* is a nickname for the second president of Yemen **منصور هادي** *mnṣwr ḥādī*. He is responsible for creating a gang called **أخونج** *āḥwnġ* affiliated with terrorism. The **أخونج** *āḥwnġ* is also a form of the previously mentioned corrup-



(a) community detection color key



(b) graph illustration after colorizing based on community detection

Figure 10.8: Graph Illustration After Colorizing Based on Community Detection, along with the Color Key

tion. *دنبوع* *dnbw* also formed the *حوثي* *hwti*. The *حوثي* *hwti* are a group who were against the government and related to military weapons. They resided in

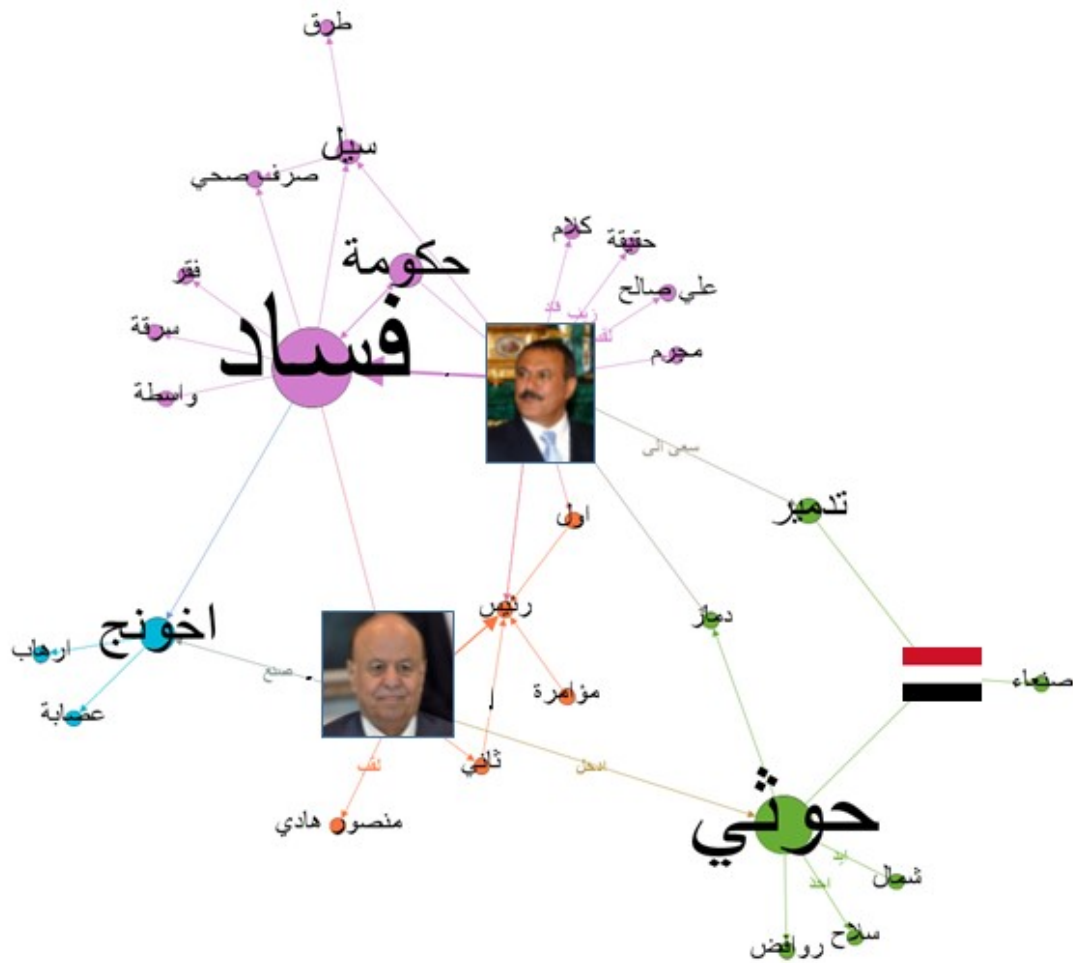


Figure 10.9: Final Resulting Graph

the north of Yemen.

We can notice, how after properly visualizing our graph, we are able to understand the main concepts of a large bulk of text. Manually going through bulks of text to retrieve information is tedious and almost impossible to achieve. Building the graphs of our data allowed us to extract information and understand the main topics of the tweet data we have.

Chapter 11

Conclusion

In conclusion, we presented in this thesis a methodology for information extraction. We were able to automate the process of extracting the important and trending topics present in social media text. To ameliorate the process of information extraction, we presented a user friendly linguistic tool that provides the morphological analysis of text. We presented the tool, ADAT, Arabic Dialect Annotation Tool and highlighted all its features. The methodology of our information extraction worked on detecting the entities and the relational entities present in the data. We evaluated our work by recording the precision, recall, and fscore, and we recorded high results. We were able to tag around 99.2% of the initial words we started with with a Fscore of 87% for *E* and 86% for *RE*. We finally constructed graphs representing our data. We analyzed the graphs and were able to draw important insights.

Appendix A

Abbreviations

ADAT	Arabic Dialect Annotation Tool
Ann	Annotation
DA	Dialect Arabic
Dist Sim	Distributional Similarity
E	Entity
G	Graph
GEO ID	Geographic Identification
IE	Information Extraction
MSA	Modern Standard Arabic
NLP	Natural Language Processing
Nb	Number
POS	Part of Speech
R	Relational Entity
Rel	Relational

Bibliography

- [1] T. K. Gupta and K. Raza, “Optimization of ann architecture: A review on nature-inspired techniques,” 2019.
- [2] “<https://cambridgeanalytica.org/>.”
- [3] “<https://www.cnet.com/news/facebook-cambridge-analytica-data-mining-and-trump-what-you-need-to-know/>.”
- [4] “<https://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/>.”
- [5] “<http://www.arabsocialmediareport.com/twitter/linechart.aspx>.”
- [6] “<http://www.un.org/en/sections/about-un/official-languages/>.”
- [7] N. Boudad, R. Faizi, R. O. H. Thami, and R. Chiheb, “Sentiment analysis in arabic: A review of the literature,” *Ain Shams Engineering Journal*, 2017.
- [8] “<http://iso639-3.sil.org/code/ara>.”
- [9] A. Pasha, M. Al-Badrashiny, M. T. Diab, A. E. 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 *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC , Reykjavik, Iceland*, pp. 1094–1101, august 2014.
- [10] S. Khalifa, N. Zalmout, and N. Habash, “YAMAMA: yet another multi-dialect arabic morphological analyzer,” in *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference System Demonstrations, Osaka, Japan*, pp. 223–227, December 2016.
- [11] H. Bouamor, N. Habash, M. Salameh, W. Zaghoulani, O. Rambow, D. Abdulrahim, O. Obeid, S. Khalifa, F. Eryani, A. Erdmann, and K. Oflazer, “The MADAR arabic dialect corpus and lexicon,” in *Conference on Language Resources and Evaluation, LREC*, 5 2018.

- [12] N. Habash and F. Eryani, “Unified guidelines and resources for arabic dialect orthography,” in *Conference on Language Resources and Evaluation, LREC*, 2018.
- [13] M. Jarrar, N. Habash, F. Alrimawi, D. Akra, and N. Zalmout, “Curras: an annotated corpus for the palestinian arabic dialect,” *Language Resources and Evaluation*, vol. 51, pp. 745–775, August 2017.
- [14] N. Habash, M. T. Diab, and O. Rambow, “Conventional orthography for dialectal arabic,” in *LREC*, pp. 711–718, 2012.
- [15] P. Stenetorp, S. Pyysalo, G. Topić, T. Ohta, S. Ananiadou, and J. Tsujii, “Brat: a web-based tool for nlp-assisted text annotation,” in *European Chapter of the Association for Computational Linguistics Demonstrations*, pp. 102–107, Association for Computational Linguistics, 2012.
- [16] T. Morton and J. LaCivita, “Wordfreak: an open tool for linguistic annotation,” in *HLT/NAACL*, 2003.
- [17] C. Müller and M. Strube, “Multi-level annotation of linguistic data with MMAX2,” *Corpus technology and language pedagogy: New resources, new tools, new methods*, 2006.
- [18] K. Dukes and N. Habash, “Morphological annotation of quranic arabic,” in *Lrec*, 2010.
- [19] I. S. Alkhazi and W. J. Teahan, “Baac: Bangor arabic annotated corpus,” *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS*, vol. 9, no. 11, pp. 131–140, 2018.
- [20] A. Abuzayed and T. Elsayed, “Quick and simple approach for detecting hate speech in arabic tweets,” in *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, pp. 109–114, 2020.
- [21] N. Taghizadeh, H. Faili, and J. Maleki, “Cross-language learning for arabic relation extraction,” *Procedia computer science*, vol. 142, pp. 190–197, 2018.
- [22] F. A. Zaraket and J. Makhlouta, “Arabic cross-document nlp for the hadith and biography literature,” in *FLAIRS Conference*, 2012.
- [23] J. Urbain, “User-driven relational models for entity-relation search and extraction,” in *Proceedings of the 1st Joint International Workshop on Entity-Oriented and Semantic Search*, p. 5, ACM, 2012.

- [24] M. B. H. Morgan and M. Van Keulen, “Information extraction for social media,” in *Proceedings of the Third Workshop on Semantic Web and Information Extraction*, pp. 9–16, 2014.
- [25] H. Trahoulsi, “Arabic named entity extraction: A local grammar-based approach,” in *Computer Science and Information Technology, 2009. IM-C SIT’09. International Multiconference on*, pp. 139–143, IEEE, 2009.
- [26] A. A. Jaber and F. A. Zaraket, “Merf: Morphology-based entity and relational entity extraction framework for arabic,” 2017.
- [27] Z. Zhang, J. Otterbacher, and D. Radev, “Learning cross-document structural relationships using boosting,” in *Proceedings of the twelfth international conference on Information and knowledge management*, pp. 124–130, ACM, 2003.
- [28] “<https://www.facebook.com/>.”
- [29] “<https://twitter.com/>.”
- [30] “<https://www.instagram.com/>.”
- [31] “<https://www.linkedin.com/>.”
- [32] F. A. Zaraket, A. Jaber, and J. Makhoulta, “Sarf: Fast and application customizable arabic morphological analyzer,” *Natural Language Engineering*, vol. 1, no. 1, pp. 1–28.
- [33] “<https://github.com/tweepy/tweepy>.”
- [34] “<https://www.un.org/en/>.”
- [35] “<http://www.almaany.com/ar/thes/ar-ar>.”
- [36] 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.
- [37] R. Řehůřek and P. Sojka, “Software Framework for Topic Modelling with Large Corpora,” in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, (Valletta, Malta), pp. 45–50, ELRA, May 2010.
- [38] I. A. Farha and W. Magdy, “Mazajak: An online arabic sentiment analyser,” in *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pp. 192–198, 2019.
- [39] “<https://pypi.org/project/graphviz/>.”

- [40] M. Bastian, S. Heymann, and M. Jacomy, “Gephi: An open source software for exploring and manipulating networks,” 2009.
- [41] M. Jacomy, “Force-atlas graph layout algorithm,” *URL: <http://gephi.org/2011/forceatlas2-the-new-version-of-our-home-brew-layout>*, 2009.
- [42] E. Voudigari, N. Salamanos, T. Papageorgiou, and E. J. Yannakoudakis, “Rank degree: An efficient algorithm for graph sampling,” in *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 120–129, IEEE, 2016.
- [43] S. K. Raghavan Unnithan, B. Kannan, and M. Jathavedan, “Betweenness centrality in some classes of graphs,” *International Journal of Combinatorics*, vol. 2014, 2014.
- [44] “<https://www.analyticsvidhya.com/blog/2020/04/community-detection-graphs-networks/>.”
- [45] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, “Fast unfolding of communities in large networks,” *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, pp. P10008–12, 2008.
- [46] “https://pypi.org/project/google_images_download/.”