CAMeL Tools: An Open Source Python Toolkit for Arabic Natural Language Processing

Ossama Obeid, Nasser Zalmout, Salam Khalifa, Dima Taji, Mai Oudah, Bashar Alhafni, Go Inoue, Fadhl Eryani, Alexander Erdmann, Nizar Habash
Computational Approaches to Modeling Language (CAMeL) Lab
New York University Abu Dhabi, UAE
{oobeid,nasser.zalmout,salamkhalifa,dima.taji,mai.oudah,
alhafni,go.inoue,fadhl.eryani,ae1541,nizar.habash}@nyu.edu

Abstract
We present CAMeL Tools, a collection of open-source tools for Arabic natural language processing in Python. CAMeL Tools currently provides utilities for pre-processing, morphological modeling, dialect identification, named entity recognition and sentiment analysis. In this paper, we describe the design of CAMeL Tools and the functionalities it provides.

Keywords: Arabic, Open Source, Morphology, Named Entity Recognition, Sentiment Analysis, Dialect Identification

1. Introduction

Over the last two decades, there have been many efforts to develop resources to support Arabic natural language processing (NLP). Some of these resources target specific NLP tasks such as tokenization, diacritization or sentiment analysis, while others attempt to target multiple tasks jointly. Most resources focus on a specific variety of Arabic such as Modern Standard Arabic (MSA), or Egyptian Arabic. These resource also vary in terms of the programming languages they use, the types of interfaces they provide, the data representations and standards they utilize, and the degree of public availability (e.g., open-source, commercial, unavailable). These variations make it difficult to write new applications that combine multiple resources.

To address many of these issues, we present CAMeL Tools, an open-source Python toolkit that supports Arabic and Arabic dialect pre-processing, morphological modeling, dialect identification, named entity recognition and sentiment analysis. CAMeL Tools provides command-line interfaces (CLIs) and application programming interfaces (APIs) covering these utilities.

The rest of the paper is organized as follows. We present some background on the difficulty of processing Arabic text (Section 2), and then discuss previous work on a variety of Arabic NLP tasks (Section 3). We describe the design and implementation of CAMeL Tools (Section 4) and provide more details about each of its components (Sections 6 to 9). Finally, we discuss some future additions to CAMeL Tools (Section 10).

2. Arabic Linguistics Background

Aside of obvious issues such as resource poverty, Arabic poses a number of challenges to NLP: orthographic ambiguity, morphological richness, dialectal variations, and orthographic noise (Habash, 2010). While these are not necessarily unique issues to Arabic, their combination makes Arabic processing particularly challenging.

Orthographic Ambiguity Arabic is generally written using the Arabic Abjad script which uses optional diacritical marks for short vowels and consonantal gemination. While the missing diacritics are not a major challenge to literate native adults, their absence is the main source of ambiguity in Arabic NLP.

Morphological Richness Arabic has a rich inflectional morphology system involving gender, number, person, aspect, mood, case, state and voice, in addition to the cliticization of a number of pronouns and particles (conjunctions, prepositions, definite article, etc.). This richness leads to MSA verbs with upwards of 5,400 forms.

Dialectal Variation While MSA is the official language of culture and education in the Arab World, it is no one's native language. A number of different local dialects (such as Egyptian, Levantine, and Gulf) are the de facto daily languages of the Arab World – both off-line and on-line. Arabic dialects differ significantly in terms of their phonology, morphology and lexicon from each other and from MSA to the point that using MSA tools and resources for processing dialects is not sufficient. For example, Khalifa et al. (2016a) report that using a state-of-the-art tool for MSA morphological disambiguation on Gulf Arabic returns POS tag accuracy at about 72%, compared to the performance on MSA, which is 96% (Pasha et al., 2014).

Orthographic Inconsistency MSA and Arabic dialects, as encountered online, show a lot of spelling inconsistency. Zaghouani et al. (2014) report that 32% of words in MSA comments online have spelling errors. Habash et al. (2018) presented 27 encountered ways to write an Egyptian Arabic word meaning 'he does not say it': e.g., مبيقولهاش *mbyqwlhAš*[1] (≈ 26,000 times), مبيقلهاش *mbyqlhAš* (≈ 1,000), and مبؤلهاش *mbŵlhAš* (less than 10). Habash et al. (2018) proposed a conventional orthography for dialectal Arabic (CODA), a set of guidelines for consistent spelling of Arabic dialects for NLP. In addition, dialectal Arabic text is also known to appear on social media in a non-standard romanization called Arabizi (Darwish, 2014).

[1] Arabic transliteration is presented in the Habash-Soudi-Buckwalter scheme (Habash et al., 2007).

(a)	**Input**	الحمد لله إيطاليا خرجت من تصفيات كأس العالم ، أظن دى فرصتنا ومالنـــاش حجة… *AlHmd llh ǍyTAlyA xrjt mn tSfyAt kÂs AlςAlm, Aðn dý frStnA w mAln___Aš Hjħ...*				
(b)	**English**	Thank God Italy is out of the world cup, I think this is our chance and we do not have an excuse				
(c)	**Preprocessing**	AlHmd llh AyTAlyA xrjt mn tSfyAt kAs AlEAlm , AZn dy frStnA wmAlnA$ Hjh ...				
(d)	**Morphological Analysis** فرصتنا *frStnA*	**Diac**	*furSitnA* فُرصِتنَا	*faraS∼atnA* فَرَصَّتنَا	*furSatunA* فُرصَتُنَا	
		Gloss	Opportunity	Compress	Holiday	
		Lemma	*furSaħ* فُرصَة	*raS∼* رَصّ	*furSaħ* فُرصَة	
		D3 Tokenization	*furSaħ +nA* فُرصَة +نَا	*fa+ raS∼at +nA* فَ+ رَصَّت +نا	*furSaħu +nA* فُرصَةُ +نَا	...
		POS	noun	verb	noun	
		Dialect	Egyptian	MSA	MSA	
(e)	**Disambiguation**		✓			
(f)	**Sentiment**	positive				
(g)	**NER**	LOC: إيطاليا (*ǍyTAlyA* ‘Italy’)				
(h)	**Dialect ID**	Cairo: 95.0%, Doha: 1.9%, Jeddah: 1.5%, Aswan: 0.8%, Khartoum: 0.3%, Other: 0.5%, MSA: 0.0%				

Table 1: An example showing the outputs of the different CAMeL Tools components

The above phenomena lead to highly complex readings that vary across many dimensions. Table 1(d) shows three analyses with different lemma readings and features for the word فرصتنا *frStnA*. Two of the three readings have the same POS (noun) but vary in terms of the English gloss and dialect. The third reading, a verb, has a different meaning, morphological analysis and tokenization.

3. Previous Work

Arabic NLP, and NLP in general, involves many tasks, some of which may be related to others. Tasks like morphological modeling, parsing, tokenization, segmentation, lemmatization, stemming and named entity recognition (NER) tend to serve as building block for higher level applications such as translation, spelling correction, and question-answering systems. Other tasks, such as dialect identification (DID) and sentiment analysis (SA), aim to compute different characteristics of input text either for analytical purposes or as additional input for some of the previously mentioned tasks.

In this section we discuss those tasks currently addressed by CAMeL Tools. We also discuss some of the software suites that have inspired the design of CAMeL Tools.

3.1. NLP Tasks

We briefly survey the different efforts for a variety of Arabic NLP tasks. Specifically, we discuss those tasks that are relevant to CAMeL Tools at the time of publishing. These include morphological modeling, NER, DID and SA.

Morphological Modeling Efforts on Arabic morphological modeling have varied over a number of dimensions, ranging from very abstract and linguistically rich representations paired with word-form derivation rules (Beesley et al., 1989; Beesley, 1996; Habash and Rambow, 2006; Smrž, 2007) to pre-compiled representations of the different components needed for morphological analysis (Buckwalter, 2002; Graff et al., 2009; Habash, 2007). Previous efforts also show a wide range for the depth of information that morphological analyzers can produce, with very shallow analyzers on one end (Vanni and Zajac, 1996), to analyzers producing form-based, functional, and lexical features as well as morpheme forms (Buckwalter, 2002; Smrž, 2007; Boudlal et al., 2010; Alkuhlani and Habash, 2011; Boudchiche et al., 2017).

Systems such as MADA (Habash et al., 2009) and MADAMIRA (Pasha et al., 2014) disambiguate the analyses that are produced by a morphological analyzer. Zalmout and Habash (2017) outline a neural model that follows the same approach. Alternatively, Farasa (Abdelali et al., 2016) relies on probabilistic models to produce high tokenization accuracy, and YAMAMA (Khalifa et al., 2016b) uses the same approach to produce MADAMIRA-like disambiguated analyses. For Arabic dialects, Zalmout et al. (2018) present a neural system that does morphological tagging and disambiguation for Egyptian Arabic.

Named Entity Recognition While some Arabic NER systems have used rule-based approaches (Shaalan and Raza, 2009; Zaghouani, 2012; Aboaoga and Aziz, 2013), most recent Arabic NER systems integrate machine learning in their architectures (Benajiba and Rosso, 2008; AbdelRahman et al., 2010; Mohammed and Omar, 2012; Oudah and Shaalan, 2012; Darwish, 2013; El Bazi and Laachfoubi, 2019). Benajiba and Rosso (2007) developed ANERsys, one of the earliest NER systems for Arabic. They built their own linguistic resources, which have become standard in Arabic NER literature: ANERcorp (i.e., an annotated corpus for Person, Location and Organization names) and ANERgazet (i.e., Person, Location and Organization gazetteers). In CAMeL Tools, we make use of all publicly available training data and compare our system to the work of Benajiba and Rosso (2008).

Dialect Identification Salameh et al. (2018) introduced a fine-grained DID system covering the dialects of 25 cities from several countries across the Arab world. Elfardy and Diab (2012) presented a set of guidelines for token-level identification of dialectness. They later proposed a supervised approach for identifying whether a given sentence is prevalently MSA or Egyptian (Elfardy and Diab, 2013) using the Arabic Online Commentary Dataset (Zaidan and

	CAMeL Tools	**MADAMIRA**	**Stanford CoreNLP**	**Farasa**
Suite Type	Arabic Specific	Arabic Specific	Multilingual	Arabic Specific
Language	Python	Java	Java with Python bindings	Java
CLI	✓	✓	✓	✓
API	✓	✓	✓	✓
Exposed Pre-processing	✓			
Morphological Modeling	✓	✓		
Morphological Disambiguation	✓	✓	✓	✓
POS Tagging	✓	✓	✓	✓
Diacritization	✓	✓		✓
Tokenization/Segementation/Stemming	✓	✓	✓	✓
Lemmatization	✓	✓		✓
Named Entity Recognition	✓	✓	✓	✓
Sentiment Analysis	✓			
Dialect ID	✓			
Parsing	Work in progress		✓	✓

Table 2: Feature comparison of CAMeL Tools, MADAMIRA, Stanford CoreNLP and Farasa.

Callison-Burch, 2011). Their system (Elfardy and Diab, 2012) combines a token-level DID approach with other features to train a Naive-Bayes classifier. Sadat et al. (2014) presented a bi-gram character-level model to identify the dialect of sentences in the social media context among dialects of 18 Arab countries. More recently, discriminating between Arabic dialects has been the goal of a dedicated shared task (Zampieri et al., 2017), encouraging researchers to submit systems to recognize the dialect of speech transcripts along with acoustic features for dialects of four main regions: Egyptian, Gulf, Levantine and North African, in addition to MSA. The dataset used in these tasks is different from the dataset we use in this work in its genre, size and the dialects covered.

Sentiment Analysis Arabic SA is a well studied problem that has many proposed solutions. These solutions span a wide array of methods such as developing lexicon-based conventional machine learning models (Badaro et al., 2014; Abdul-Mageed and Diab, 2014) or deep learning models (Abu Farha and Magdy, 2019; Badaro et al., 2018; Baly et al., 2017). More recently, fine-tuning large pre-trained language models has achieved state-of-the-art results on various NLP tasks. ElJundi et al. (2019) developed a universal language model for Arabic (hULMonA) which was pre-trained on a large corpus of Wikipedia sentences and compared its performance to multilingual BERT (mBERT) (Devlin et al., 2018) on the task of Arabic SA after fine-tuning. They have shown that although mBERT was only trained on Modern Standard Arabic (MSA), it can still achieve state-of-the-art results on some datasets if it's fine-tuned on dialectal Arabic data. Furthermore, Antoun et al. (2020) pre-trained an Arabic specific BERT (AraBERT) and were able to achieve state-of-the-art results on several downstream tasks.

3.2. Software Suites

While most efforts focus on individual tasks, there are few that provide multiple capabilities in the form of a unified toolkit. These can be classified as Arabic specific, such as MADAMIRA (Pasha et al., 2014) and Farasa (Abdelali et al., 2016; Darwish and Mubarak, 2016) or multi-lingual, such as Stanford CoreNLP (Manning et al., 2014). Below we discuss the capabilities of the mentioned tools and we provide a rough comparison in Table 2.

MADAMIRA provides a single, yet configurable, mode of operation revolving around it's morphological analyzer. When disambiguation is enabled, features such as POS tagging, tokenization, segmentation, lemmatization, NER and base-phrase chunking become available. MADAMIRA was designed specifically for Arabic and supports both MSA and Egyptian and primarily provides a CLI, a server mode, and a Java API.

Farasa is a collection of Java libraries and CLIs for MSA.[2] These include separate tools for diacritization, segmentation, POS tagging, parsing, and NER. This allows for a more flexible usage of components compared to MADAMIRA.

Stanford CoreNLP is a multilingual Java library, CLI and server providing multiple NLP components with varying support for different languages. Arabic support is provided for parsing, tokenization, POS tagging, sentence splitting and NER (a rule-based system using regular expressions). An official Python library is also available (Qi et al., 2018) which provides official bindings to CoreNLP as well as providing neural implementations for some of their components. It is worth noting that the commonly used Natural Language Toolkit (NLTK) (Loper and Bird, 2002) provides Arabic support through unofficial bindings to Stanford CoreNLP.

4. Design and Implementation

CAMeL Tools is an open-source package consisting of a set of Python APIs with accompanying command-line tools that are thin wrap these APIs. In this section we discuss the design decisions behind CAMeL Tools and how it was implemented.

4.1. Motivation

There are two main issues with currently available tools that prompted us to develop CAMeL Tools and influenced its design:

[2] http://qatsdemo.cloudapp.net/farasa/

Fragmented Packages and Standards Tools tend to focus on one task with no easy way to glue together different packages. Some tools do provide APIs, making this easier, but others only provide command-line tools. This leads to overhead writing glue code including interfaces to different packages and parsers for different output formats.

Lack of Flexibility Most tools don't expose intermediate functionality. This makes it difficult to reuse intermediate output to reduce redundant computation. Additionally, functions for pre-processing input don't tend to be exposed to users which makes it difficult to replicate any pre-processing performed internally.

4.2. Design Philosophy

Here we identify the driving principles behind the design of CAMeL Tools. These have been largely inspired by the designs of MADAMIRA, Farasa, CoreNLP and NLTK, and include our own personal requirements.

Arabic Specific We want a toolkit that focuses solely on Arabic NLP. Language agnostic tools don't tend to model the complexities of Arabic text very well, or indeed, other morphologically complex languages. Doing so would increase software complexity and development time.

Flexibility Provide flexible components to allow mixing and matching rather than providing large monolithic applications. We want CAMeL Tools to be usable by both application developers requiring only high-level access to components, and researchers, who might want to experiment with new implementations of components or their sub-components.

Modularity Different implementations of the same component type should be slot-in replacements for one another. This would allow for easier experimentation as well as allowing users to provide custom implementations as a replacement for built-in components.

Performance Components should provide close to state-of-the-art results but within reasonable run-time performance and memory usage. However, we don't want this to come at the cost of code readability and maintainability and we are willing to sacrifice some performance benefits to this end.

Ease of Use Using individual components shouldn't take more than a few lines of code to get started with. We aim to provide meaningful default configurations that are generally applicable in most situations.

4.3. Implementation

CAMeL Tools is implemented in Python 3 and can be installed through pip.[3] We chose Python due to its ease of use and its pervasiveness in NLP and Machine Learning along with libraries such as tensorflow, pytorch, and scikit-learn. We aim to be compatible with Python version 3.5 and later running on Linux, MacOS, and Windows. CAMeL Tools is in continuous development with new features being added and older ones being improved. As such, it is difficult to accurately report on the performance of each component. In this paper, we will report the state of CAMeL Tools at the time of publishing and updates will be published on dedicated web page.[4]

At the moment, CAMeL Tools provides utilities for pre-processing, morphological analysis and disambiguation, DID, NER and SA. In the following sections, we discuss these components in more detail.

5. Pre-processing Utilities

CAMeL Tools provides a set of pre-processing utilities that are common in Arabic NLP but get re-implemented frequently. Different tools and packages tend to do slightly different pre-processing steps that are often not well documented or exposed for use on their own. By providing these utilities as part of the package, we hope to reduce the overhead of writing Arabic NLP applications and insure that pre-processing is consistent from one project to the other. In this section, we discuss the different pre-processing utilities that come with CAMeL Tools.

Transliteration When working with Arabic text, it is sometimes convenient or even necessary to use alternate transliteration schemes. Buckwalter transliteration (Buckwalter, 2002) and its variants, Safe Buckwalter and XML Buckwalter, for example, map the Arabic character set into ASCII representations using a one-to-one map. These transliterations are used as input or output formats in Arabic NLP tools such as MADAMIRA (Pasha et al., 2014), and as the chosen encoding of various resources such as the SAMA database (Graff et al., 2009). Habash-Soudi-Buckwalter (HSB) (Habash et al., 2007) is another variant on the Buckwalter transliteration scheme that includes some non-ASCII characters whose pronunciation is easier to remember for non-Arabic speakers. We provide transliteration functions to and from the following schemes: Arabic script, Buckwalter, Safe Buckwalter, XML Buckwalter, and Habash-Soudi-Buckwalter. Additionally, we provide utilities for users to define their own transliteration schemes.

Orthographic Normalization Due to Arabic's complex morphology, it is necessary to normalize text in various ways in order to reduce noise and sparsity (Habash, 2010). The most common of these normalizations are:

- **Unicode normalization** which includes breaking up combined sequences (e.g. ﻻ to ل and ا), converting character variant forms to a single canonical form (e.g. ﻊ, ﻋ, and ﻌ to ع), and converting extensions to the Arabic character set used for Persian and Urdu to the closest Arabic character (e.g. گ to ك).

- **Dediacritization** which removes Arabic diacritics which occur infrequently in Arabic text and tend to be considered noise. These include short vowels, shadda (gemination marker), and the dagger alif (e.g. مُدَرِّسَةُ *mudar∼isaħu* to مدرسة *mdrsħ*).

[3]Installation instructions and documentation can be found on `https://github.com/CAMeL-Lab/camel_tools`

[4]`https://camel-lab.github.io/camel_tools_updates/`

- **Removal of unnecessary characters** including those with no phonetic value, such as the tatweel (kashida) character.

- **Letter variant normalization** for letters that are so often misspelled. This includes normalizing all the forms of Hamzated Alif (أ $\hat{A}$, إ $\check{A}$, آ $\bar{A}$) to bare Alif (ا A), the Alif-Maqsura (ى $\acute{y}$) to Ya (ي y), the Ta-Marbuta (ة $\hbar$) to Ha (ه h), and the non-Alif forms of Hamza (ؤ $\hat{w}$, ئ $\hat{y}$) to the Hamza letter (ء ’).

Table 1(c), illustrates how these pre-processing utilities can be applied to an Arabic sentence. Specifically, we apply Arabic to Buckwalter transliteration, letter variant normalization, and tatweel removal.

6. Morphological Modeling

Morphology is one of the most studied aspects of the Arabic language. The richness of Arabic makes the modeling of morphology a complex task, justifying the focus put on morphological analysis functionalities.

6.1. Analysis, Generation, and Reinflection

As part of CAMeL Tools, we currently provide implementations of the CALIMA$_{Star}$ analyzer, generator, and reinflector described by Taji et al. (2018). These components operate on databases that are in the same format as the ALMORGEANA database (Habash, 2007), which extends the BAMA databases (Buckwalter, 2002) comprising three lexicon tables for prefixes, suffixes, and stems, and three compatibility tables for prefix-suffix, prefix-stem, and stem-suffix. CAMeL Tools include databases for MSA as well as the Egyptian and Gulf dialects.

Analysis For our tool's purposes, we define analysis as the identification of all the different readings (analyses) of a word out of context. Each of these readings is defined by a lexical features such as lemma, gloss, and stem, and morphological features such POS tags, gender, number, case, and mood. The analyzer expects a word for input, and tries to match a prefix, stem, and suffix from the database lexicon table to the surface form of the word, while maintaining the constraints imposed by the compatibility tables. This follows the algorithm described by Buckwalter (2002) with some enhancements. These enhancements, along with the switch of encoding to Arabic script, allows our analyzer to better detect and handle foreign words, punctuation, and digits.

Generation Generation is the task of inflecting a lemma for a set of morphological features. The algorithm we use for generation follow the description of the generation component in ALMORGEANA (Habash, 2007). The generator expects minimally a lemma, and a POS tag, and produces a list of inflected words with their full analysis. If inflectional features, such as person, gender, and case, are specified in the input, we limit the generated output to those values. We generate all possible values for the inflectional features that are not specified. Clitics, being optional features, are only generated when they are specified.

Reinflection Reinflection differs from generation in that the input to the reinflector is a fully inflected word, with a list of morphological feature. The reinflector is not limited to a specific lemma or POS tag, but rather uses the analyzer component to generate all possible analyses that the input word has, including the possible lemmas and POS tags. Next, the reinflector uses the list of features it had for input, along with the features from the analyzer, to create a new list of features that can be input to the generator to produce a reinflection for every possible analysis of the input word.
In table 1(d), we present an example of the morphological analysis which CAMeL Tools can provide. The analysis includes but not limited to diacritization, the CAMEL Arabic Phonetic Inventory (CAPHI), English gloss, tokenization, lemmatization, and POS tagging.

6.2. Disambiguation and Annotation

Morphological disambiguation is the task of choosing an analysis of a word in context. Tasks such as diacritization, POS tagging, tokenization and lemmatization can be considered forms of disambiguation. When dealing with Arabic however, each of these tasks on their own, don't fully disambiguate a word. In order to avoid confusion, we refer to the individual tasks on their own as annotation tasks. Traditionally, annotation tasks would be implemented independently of each other. We however, chose to use the one-fell-swoop approach of Habash and Rambow (2005). This approach entails predicting a set of features of a given word, ranking its out-of-context analyses based on the predicted features, and finally choosing the top ranked analysis. Annotation can then be achieved by retrieving the relevant feature from the ranked analysis.
We provide two builtin disambiguators: a simple low-cost disambiguator based on a maximum likelihood estimation (MLE) model and a neural network disambiguator that provides improved disambiguation accuracy using multitask learning. Table 1(e) shows how these morphological disambiguators can disambiguate a word.

MLE Disambiguator We built a simple disambiguation model based on YAMAMA (Khalifa et al., 2016b), where the main component is a lookup model of a word and its most probable full morphological analysis. For a word that does appear in the lookup model, we use the morphological analyzer from Section 6.1 to generate all possible analyses for the word and chose the top ranked analysis based on the pre-computed probabilistic score for the joint lemma and POS frequency. Both the lookup model and the pre-computed scores are based on the same training dataset for the given dialect.

Multitask Learning Disambiguator We provide a simplified implementation of the neural multitask learning approach to disambiguation by Zalmout and Habash (2019). Instead of the LSTM-based (long short-term memory) language models used in the original system, we use unigram language models without sacrificing much accuracy. This is done to increase run-time performance and decrease memory usage.
We evaluated the CAMeL Tools disambiguators against

		CAMeL Tools	
MSA	MADAMIRA	Multitask	MLE
DIAC	87.7%	**90.9%**	78.4%
LEX	**96.4%**	95.4%	95.7%
POS	97.1%	**97.2%**	95.5%
FULL	85.6%	**89.0%**	70.0%
ATB TOK	99.0%	**99.4%**	99.0%

Table 3: Comparison of the performance of the CAMeL Tools Multitask learning and MLE systems on MSA to MADAMIRA. The systems are evaluated on their accuracy to correctly predict diacritics (DIAC), lemmas (LEX), part-of-speech (POS), the full set of predicted features (FULL), and the ATB tokenization.

		CAMeL Tools
EGY	MADAMIRA	MLE
DIAC	**82.8%**	78.9%
LEX	86.6%	**87.8%**
POS	91.7%	**91.8%**
FULL	**76.4%**	73.0%
BW TOK	**93.5%**	92.8%

Table 4: Comparison of the performance of the CAMeL Tools MLE systems on Egyptian to MADAMIRA. The systems are evaluated on their accuracy to correctly predict diacritics (DIAC), lemmas (LEX), part-of-speech (POS), the full set of predicted features (FULL), and the Buckwalter tag tokenization (BW TOK) (Khalifa et al., 2016b).

MADAMIRA.[5] For the accuracy evaluation we used the Dev sets recommended by Diab et al. (2013) of the Penn Arabic Treebank (PATB parts 1,2 and 3) (Maamouri et al., 2004) for MSA and the ARZ Treebank (Maamouri et al., 2014) for EGY.

Table 3 provides the evaluation on MSA on MADAMIRA using the SAMA database (Graff et al., 2009), CAMeL Tools multitask learning system,[6] and CAMeL Tools MLE system. Table 4 provides the evaluation on EGY on MADAMIRA using the CALIMA ARZ database (Habash et al., 2012), following the work described by Pasha et al. (2015) and Khalifa et al. (2016b), and the CAMeL Tools MLE system.

We compare to Farasa's tokenizer (Abdelali et al., 2016), and the accuracy of their system's ATB tokenization on the same MSA Dev set was 98%. However, this number is not fair to report for the sake of this evaluation because Farasa follows slightly different tokenization conventions. For example, the Ta (ت *t*) in words such as مدرستها *mdrsthA* 'her-school' is not reverted to its original Ta-Marbuta (ة *ħ*) form, unlike the convention we have adopted for ATB tokenization.

[5]MADAMIRA: Released on April 03, 2017, version 2.1

[6]The implementation we report on is an older version that uses TensorFlow 1.8. We are currently working on a new implementation using TensorFlow 2.1.

7. Dialect Identification

We provide an implementation of the Dialect Identification (DID) system described by Salameh et al. (2018). This system is able to distinguish between 25 Arabic city dialects and MSA. Table 1(h), shows an example of the our system's output.

Approach We train a Multinomial Naive Bayes (MNB) classifier that outputs 26 probability scores referring to the 25 cities and MSA. The model is fed with a suite of features covering word unigrams and character unigrams, bigrams and trigrams weighted by their Term Frequency-Inverse Document Frequency (TF-IDF) scores, combined with word-level and character-level language model scores for each dialect. We also train a secondary MNB model using this same setup trained that outputs six probability scores referring to five city dialects and MSA for which the MADAR corpus provides additional training data. The output of the secondary model is used as additional features to the primary model.

Datasets We train our system using the MADAR corpus (Bouamor et al., 2018), a large-scale collection of parallel sentences built to cover the dialects of 25 cities from the Arab World, in addition to MSA. It is composed of two sub-corpora; the first consisting of 1,600 training sentences and covering all 25 city dialects and MSA, and the other consisting of 9,000 training sentences and covering five city dialects and MSA. The first sub-corpus is used to train our primary model while the latter is used to train our secondary model. We use the full corpus to train our language models.

Experiments and Results We evaluate our system using the test split of the MADAR corpus. Our system is able to distinguish between the 25 Arabic city dialects and MSA with an accuracy of 67.9% for sentences with an average length of 7 words with an accuracy of 90% for sentences consisting of 16 words. Our DID system is used in the back-end component of ADIDA (Obeid et al., 2019) to compute the dialect probabilities of a given input and display them as point-map or a heat-map on top of a geographical map of the Arab world.

8. Named Entity Recognition

We use a large pre-trained language model to build an Arabic named entity recognition system targeting four classes: Location (LOC), Miscellaneous (MISC), Organization (ORG), and Person (PERS). Table 1(g) shows an example of the output of CAMeL Tools NER on an input Arabic sentence.

Approach We used HuggingFace's Transformers (Wolf et al., 2019) to fine-tune AraBERT (Antoun et al., 2020) for labeling named entities in the commonly used IOB (inside, outside, beginning) NER tagging format. The fine-tuning was done by adding a fully connected linear layer with a softmax activation function to the last hidden state. We use the representation of the first sub-token as an input to the linear layer. We report results on the fine-tuned model.

Datasets We train on the publicly available Arabic NER dataset ANERcorp ($\sim$150K words) (Benajiba et al., 2007). Since the exact ANERcorp splits of training and test are not

	CAMeL Tools			Benajiba&Rosso'08		
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**
LOC	88%	**92%**	**90%**	**93%**	87%	**90%**
MISC	68%	**58%**	**63%**	**71%**	54%	61%
ORG	77%	**70%**	**73%**	**84%**	54%	66%
PERS	**89%**	**85%**	**87%**	80%	67%	73%
Overall	84%	**81%**	**83%**	**87%**	73%	79%

Table 5: The results of the proposed system when trained and tested on ANERcorp dataset vs. CRF-based system (Benajiba and Rosso, 2008).

available, we followed the same split ratios they used: 5/6 for training and 1/6 for testing. We will make our exact splits available for the wider community.[7]

Experiments and Results AraBERT was fine-tuned on a single GPU for three epochs with a learning rate of 5e-05, batch size of 32, and a maximum sequence length of 256. We report our results in Table 5 in terms of entity-based precision, recall and F1 score, using the CoNLL script for NER evaluation. We compare to Benajiba and Rosso (2008)'s CRF-based system. Overall, our system improves over their results in terms of recall and F1 score.

9. Sentiment Analysis

We leverage large pre-trained language models to build an Arabic sentiment analyzer as part of CAMeL Tools. Our analyzer classifies an Arabic sentence into being positive, negative, or neutral. Table 1(f), shows an example of how CAMeL Tools sentiment analyzer takes an Arabic sentence as an input and outputs its sentiment.

Approach We used HuggingFace's Transformers (Wolf et al., 2019) to fine-tune multilingual BERT (mBERT) (Devlin et al., 2018) and AraBERT (Antoun et al., 2020) on the task of Arabic SA. The fine-tuning was done by adding a fully connected linear layer with a softmax activation function to the last hidden state. We report results on both fine-tuned models and we provide the best of the two as part of CAMeL Tools.

Datasets To ensure that mBERT and AraBERT can be generalized to classify the sentiment of dialectal tweets, we used various datasets for fine-tuning and evaluation. The first dialectal dataset is the Arabic Speech-Act and Sentiment Corpus of Tweets (ArSAS) (Elmadany et al., 2018) where each tweet is annotated for positive, negative, neutral, and mixed sentiment. The second dataset is the Arabic Sentiment Tweets Dataset (ASTD) (Nabil et al., 2015) that is in both Egyptian Arabic and MSA. Each tweet has a sentiment label of either positive, negative, neutral, or objective. The third dataset is SemEval-2017 task 4-A benchmark dataset (Rosenthal et al., 2017) which is in MSA and each tweet was annotated with a sentiment label of positive, negative, or neutral. Lastly, we used the Multi-Topic Corpus for Target-based Sentiment Analysis in Arabic Levantine Tweets (ArSenTD-Lev) (Baly et al., 2019) where tweet was labeled as being positive, very positive, neutral, negative, or very negative. All the tweets in the datasets were pre-processed using the utilities provided by CAMeL Tools to remove diacritics, URLs, and usernames.

[7]Data split details are linked from `https://camel-lab.github.io/camel_tools_updates/`

	CAMeL Tools		
	AraBERT	**mBERT**	**Mazajak**
ArSAS	**92%**	89%	90%
ASTD	**73%**	66%	72%
SemEval	**69%**	60%	63%

Table 6: CAMeL Tools sentiment analyzer performance using AraBERT and mBERT compared to Mazajak over three benchmark datasets. The results are reported in terms of macro F1 score over the positive and negative classes.

Experiments and Results We compare our results to Mazajak (Abu Farha and Magdy, 2019) and we tried to follow their approach in terms of evaluation and experimental setup. We discarded the objective tweets from the ASTD dataset and applied 80/20 random sampling to split the dataset into train/test respectively. We also discarded the mixed class from the ArSAS dataset and kept the tweets with an annotation confidence of over 50% and applied 80/20 random sampling to split the dataset into train/test respectively. Additionally, we reduced the number of labels in the ArSenTD-Lev dataset to positive, negative, and neutral by turning the very positive labels to positive and the very negative labels to negative. We will make our exact splits available for the wider community.[7]
Both mBERT and AraBERT were fine-tuned on ArSenTD-Lev and the train splits from SemEval, ASTD, and ArSAS (24,827 tweets) on a single GPU for 3 epochs with a learning rate of 3e-5, batch size of 32, and a maximum sequence length of 128. For evaluation, we use the F_1^{PN} score which was defined by SemEval-2017 and used by Majazak; F_1^{PN} is the macro F1 score over the positive and negative classes only while neglecting the neutral class. Table 6 shows our results compared to Mazajak.

10. Conclusion and Future Work

We presented CAMeL Tools, an open source set of tools for Arabic NLP providing utilities for pre-processing, morphological modeling, dialect identification, named entity recognition and sentiment analysis.
CAMeL Tools is in a state of active development. We will continue to add new API components and command-line tools while also improving existing functionality. Additionally, we will be adding more datasets and models to support more dialects. Some features we are currently working on adding to CAMeL Tools include:

- A transliterator to and from Arabic script and Arabizi inspired by Al-Badrashiny et al. (2014).
- A spelling correction component supporting MSA and DA text inspired by the work of Watson et al. (2018) and Eryani et al. (2020).
- Additional morphological disambiguators for Arabic dialects, building on work by Khalifa et al. (2020).

- A dependency parser based on the CAMeL Parser by Shahrour et al. (2016).
- More Dialect ID options including more cities and a new model for country/region based identification.

Acknowledgments

The support and resources from the High Performance Computing Center at New York University Abu Dhabi are gratefully acknowledged.

Bibliographical References

Abdelali, A., Darwish, K., Durrani, N., and Mubarak, H. (2016). Farasa: A fast and furious segmenter for Arabic. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 11–16, San Diego, California.

AbdelRahman, S., Elarnaoty, M., Magdy, M., and Fahmy, A. (2010). Integrated machine learning techniques for Arabic named entity recognition. *International Journal of Computer Science Issues (IJCSI)*, 7:27–36.

Abdul-Mageed, M. and Diab, M. (2014). SANA: A large scale multi-genre, multi-dialect lexicon for Arabic subjectivity and sentiment analysis. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, pages 1162–1169, Reykjavik, Iceland.

Aboaoga, M. and Aziz, M. J. A. (2013). Arabic person names recognition by using a rule based approach. *Journal of Computer Science*, 9:922–927.

Abu Farha, I. and Magdy, W. (2019). Mazajak: An online Arabic sentiment analyser. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 192–198, Florence, Italy, August. Association for Computational Linguistics.

Al-Badrashiny, M., Eskander, R., Habash, N., and Rambow, O. (2014). Automatic transliteration of romanized Dialectal Arabic. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*, pages 30–38, Ann Arbor, Michigan.

Alkuhlani, S. and Habash, N. (2011). A Corpus for Modeling Morpho-Syntactic Agreement in Arabic: Gender, Number and Rationality. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*, Portland, Oregon, USA.

Antoun, W., Baly, F., and Hajj, H. (2020). AraBERT: Transformer-based model for Arabic language understanding.

Badaro, G., Baly, R., Hajj, H., Habash, N., and El-Hajj, W. (2014). A large scale Arabic sentiment lexicon for Arabic opinion mining. In *Proceedings of the Workshop for Arabic Natural Language Processing (WANLP)*, pages 165–173, Doha, Qatar.

Badaro, G., El Jundi, O., Khaddaj, A., Maarouf, A., Kain, R., Hajj, H., and El-Hajj, W. (2018). EMA at SemEval-2018 task 1: Emotion mining for Arabic. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 236–244, New Orleans, Louisiana, June. Association for Computational Linguistics.

Baly, R., Hajj, H., Habash, N., Shaban, K. B., and El-Hajj, W. (2017). A sentiment treebank and morphologically enriched recursive deep models for effective sentiment analysis in Arabic. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 16(4):23.

Baly, R., Khaddaj, A., Hajj, H., El-Hajj, W., and Shaban, K. B. (2019). ArSentD-LEV: A multi-topic corpus for target-based sentiment analysis in Arabic Levantine tweets.

Beesley, K., Buckwalter, T., and Newton, S. (1989). Two-Level Finite-State Analysis of Arabic Morphology. In *Proceedings of the Seminar on Bilingual Computing in Arabic and English*.

Beesley, K. (1996). Arabic Finite-State Morphological Analysis and Generation. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 89–94, Copenhagen, Denmark.

Benajiba, Y. and Rosso, P. (2007). ANERsys 2.0: Conquering the ner task for the Arabic language by combining the maximum entropy with pos-tag information. In *Proceedings of workshop on natural language-independent engineering, 3rd Indian international conference on artificial intelligence (IICAI-2007)*, pages 1814–1823.

Benajiba, Y. and Rosso, P. (2008). Arabic Named Entity Recognition using Conditional Random Fields. In *Proceedings of workshop on HLT NLP within the Arabic world (LREC)*.

Benajiba, Y., Rosso, P., and Benedí Ruiz, J. M. (2007). ANERsys: An Arabic Named Entity Recognition System Based on Maximum Entropy. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, pages 143–153, Berlin, Heidelberg. Springer Berlin Heidelberg.

Bouamor, H., Habash, N., Salameh, M., Zaghouani, W., Rambow, O., Abdulrahim, D., Obeid, O., Khalifa, S., Eryani, F., Erdmann, A., and Oflazer, K. (2018). The MADAR Arabic Dialect Corpus and Lexicon. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Miyazaki, Japan.

Boudchiche, M., Mazroui, A., Bebah, M. O. A. O., Lakhouaja, A., and Boudlal, A. (2017). AlKhalil Morpho Sys 2: A robust Arabic morpho-syntactic analyzer. *Journal of King Saud University - Computer and Information Sciences*, 29(2):141–146.

Boudlal, A., Lakhouaja, A., Mazroui, A., Meziane, A., Bebah, M., and Shoul, M. (2010). Alkhalil Morpho Sys1: A morphosyntactic analysis system for Arabic texts. In *Proceedings of the International Arab Conference on Information Technology*, pages 1–6.

Buckwalter, T. (2002). Buckwalter Arabic morphological analyzer version 1.0. Linguistic Data Consortium (LDC) catalog number LDC2002L49, ISBN 1-58563-257-0.

Darwish, K. and Mubarak, H. (2016). Farasa: A new fast and accurate Arabic word segmenter. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Portorož, Slovenia.

Darwish, K. (2013). Named entity recognition using cross-lingual resources: Arabic as an example. In *Proceedings of the 51st Annual Meeting of the Association for Com-*

putational Linguistics, pages 1558–1567.

Darwish, K. (2014). Arabizi Detection and Conversion to Arabic. In *Proceedings of the Workshop for Arabic Natural Language Processing (WANLP)*, pages 217–224, Doha, Qatar.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding.

Diab, M., Habash, N., Rambow, O., and Roth, R. (2013). LDC Arabic treebanks and associated corpora: Data divisions manual. *arXiv preprint arXiv:1309.5652*.

El Bazi, I. and Laachfoubi, N. (2019). Arabic named entity recognition using deep learning approach. *International Journal of Electrical and Computer Engineering*, 9:2025–2032.

Elfardy, H. and Diab, M. (2012). Simplified guidelines for the creation of large scale dialectal Arabic annotations. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Istanbul, Turkey.

Elfardy, H. and Diab, M. (2013). Sentence Level Dialect Identification in Arabic. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*, pages 456–461, Sofia, Bulgaria.

ElJundi, O., Antoun, W., El Droubi, N., Hajj, H., El-Hajj, W., and Shaban, K. (2019). hULMonA: The universal language model in Arabic. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 68–77, Florence, Italy, August. Association for Computational Linguistics.

Elmadany, A., Mubarak, H., and Magdy, W. (2018). Arsas: An Arabic speech-act and sentiment corpus of tweets. In Hend Al-Khalifa, et al., editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France, may. European Language Resources Association (ELRA).

Eryani, F., Habash, N., Bouamor, H., and Khalifa, S. (2020). A Spelling Correction Corpus for Multiple Arabic Dialects. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Marseille, France.

Graff, D., Maamouri, M., Bouziri, B., Krouna, S., Kulick, S., and Buckwalter, T. (2009). Standard Arabic Morphological Analyzer (SAMA) Version 3.1. Linguistic Data Consortium LDC2009E73.

Habash, N. and Rambow, O. (2005). Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*, pages 573–580, Ann Arbor, Michigan.

Habash, N. and Rambow, O. (2006). MAGEAD: A morphological analyzer and generator for the Arabic dialects. In *Proceedings of the International Conference on Computational Linguistics and the Conference of the Association for Computational Linguistics (COLING-ACL)*, pages 681–688, Sydney, Australia.

Habash, N., Soudi, A., and Buckwalter, T. (2007). On Arabic Transliteration. In A. van den Bosch et al., editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*, pages 15–22. Springer, Netherlands.

Habash, N., Rambow, O., and Roth, R. (2009). MADA+TOKAN: A toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization. In Khalid Choukri et al., editors, *Proceedings of the International Conference on Arabic Language Resources and Tools*, Cairo, Egypt. The MEDAR Consortium.

Habash, N., Eskander, R., and Hawwari, A. (2012). A Morphological Analyzer for Egyptian Arabic. In *Proceedings of the Workshop of the Special Interest Group on Computational Morphology and Phonology (SIGMORPHON)*, pages 1–9, Montréal, Canada.

Habash, N., Eryani, F., Khalifa, S., Rambow, O., Abdulrahim, D., Erdmann, A., Faraj, R., Zaghouani, W., Bouamor, H., Zalmout, N., Hassan, S., shargi, F. A., Alkhereyf, S., Abdulkareem, B., Eskander, R., Salameh, M., and Saddiki, H. (2018). Unified guidelines and resources for Arabic dialect orthography. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Miyazaki, Japan.

Habash, N. (2007). Arabic Morphological Representations for Machine Translation. In Antal van den Bosch et al., editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Kluwer/Springer.

Habash, N. Y. (2010). *Introduction to Arabic natural language processing*. Morgan & Claypool Publishers.

Khalifa, S., Habash, N., Abdulrahim, D., and Hassan, S. (2016a). A Large Scale Corpus of Gulf Arabic. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Portorož, Slovenia.

Khalifa, S., Zalmout, N., and Habash, N. (2016b). Yamama: Yet another multi-dialect Arabic morphological analyzer. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 223–227, Osaka, Japan.

Khalifa, S., Zalmout, N., and Habash, N. (2020). Morphological Disambiguation for Gulf Arabic: The Interplay between Resources and Methods. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Marseille, France.

Loper, E. and Bird, S. (2002). NLTK: The Natural Language Toolkit. In *Proceedings of the Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, pages 63–70, Philadelphia, Pennsylvania.

Maamouri, M., Bies, A., Buckwalter, T., and Mekki, W. (2004). The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus. In *Proceedings of the International Conference on Arabic Language Resources and Tools*, pages 102–109, Cairo, Egypt.

Maamouri, M., Bies, A., Kulick, S., Ciul, M., Habash, N., and Eskander, R. (2014). Developing an Egyptian Arabic Treebank: Impact of Dialectal Morphology on Annotation and Tool Development. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Reykjavik, Iceland.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*, pages 55–60.

Mohammed, N. and Omar, N. (2012). Arabic Named Entity Recognition using artificial neural network. *Journal of Computer Science*, 8:1285–1293.

Nabil, M., Aly, M., and Atiya, A. (2015). ASTD: Arabic sentiment tweets dataset. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2515–2519, Lisbon, Portugal, September. Association for Computational Linguistics.

Obeid, O., Salameh, M., Bouamor, H., and Habash, N. (2019). ADIDA: Automatic dialect identification for Arabic. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 6–11, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Oudah, M. and Shaalan, K. (2012). A pipeline Arabic named entity recognition using a hybrid approach. In *Proceedings of the 24th international conference on computational linguistics (COLING 2012)*, pages 2159–2176.

Pasha, A., Al-Badrashiny, M., Diab, M., Kholy, A. E., Eskander, R., Habash, N., Pooleery, M., Rambow, O., and Roth, R. (2014). MADAMIRA: A fast, comprehensive tool for morphological analysis and disambiguation of Arabic. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, pages 1094–1101, Reykjavik, Iceland.

Pasha, A., Al-Badrashiny, M., Diab, M., Habash, N., Pooleery, M., and Rambow, O., (2015). *MADAMIRA v2.0 User Manual.*

Qi, P., Dozat, T., Zhang, Y., and Manning, C. D. (2018). Universal dependency parsing from scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170, Brussels, Belgium, October. Association for Computational Linguistics.

Rosenthal, S., Farra, N., and Nakov, P. (2017). Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the International Workshop on Semantic Evaluation (SemEval)*, pages 501–516, Vancouver, Canada.

Sadat, F., Kazemi, F., and Farzindar, A. (2014). Automatic Identification of Arabic Dialects in Social Media. In *Proceedings of the Workshop on Natural Language Processing for Social Media (SocialNLP)*, pages 22–27, Dublin, Ireland.

Salameh, M., Bouamor, H., and Habash, N. (2018). Fine-grained Arabic dialect identification. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 1332–1344, Santa Fe, New Mexico, USA.

Shaalan, K. and Raza, H. (2009). Nera: Named entity recognition for Arabic. *Journal of the American Society for Information Science and Technology*, 60:1652–1663.

Shahrour, A., Khalifa, S., Taji, D., and Habash, N. (2016). CamelParser: A system for Arabic syntactic analysis and morphological disambiguation. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 228–232.

Smrž, O. (2007). ElixirFM — Implementation of Functional Arabic Morphology. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages (CASL)*, pages 1–8, Prague, Czech Republic. ACL.

Taji, D., Khalifa, S., Obeid, O., Eryani, F., and Habash, N. (2018). An Arabic Morphological Analyzer and Generator with Copious Features. In *Proceedings of the Fifteenth Workshop on Computational Research in Phonetics, Phonology, and Morphology (SIGMORPHON)*, pages 140–150.

Vanni, M. and Zajac, R. (1996). The Temple translator's workstation project. In *Proceedings of the TIPSTER Text Program Workshop*, pages 101–106, Vienna, Virginia.

Watson, D., Zalmout, N., and Habash, N. (2018). Utilizing character and word embeddings for text normalization with sequence-to-sequence models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 837–843, Brussels, Belgium, October-November. Association for Computational Linguistics.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., and Brew, J. (2019). Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Zaghouani, W., Mohit, B., Habash, N., Obeid, O., Tomeh, N., Rozovskaya, A., Farra, N., Alkuhlani, S., and Oflazer, K. (2014). Large Scale Arabic Error Annotation: Guidelines and Framework. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Reykjavik, Iceland.

Zaghouani, W. (2012). RENAR: A rule-based Arabic named entity recognition system. *ACM Transactions on Asian Language Information Processing*, 11:1–13.

Zaidan, O. F. and Callison-Burch, C. (2011). The Arabic Online Commentary Dataset: an Annotated Dataset of Informal Arabic With High Dialectal Content. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*, pages 37–41.

Zalmout, N. and Habash, N. (2017). Don't throw those morphological analyzers away just yet: Neural morphological disambiguation for Arabic. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 704–713, Copenhagen, Denmark.

Zalmout, N. and Habash, N. (2019). Adversarial multitask learning for joint multi-feature and multi-dialect morphological modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1775–1786, Florence, Italy, July. Association for Computational Linguistics.

Zalmout, N., Erdmann, A., and Habash, N. (2018). Noise-robust morphological disambiguation for dialectal Arabic. In *Proceedings of the Conference of the North Amer-*

ican Chapter of the Association for Computational Linguistics (NAACL), New Orleans, Louisiana, USA.

Zampieri, M., Malmasi, S., Ljubešić, N., Nakov, P., Ali, A., Tiedemann, J., Scherrer, Y., and Aepli, N. (2017). Findings of the VarDial Evaluation Campaign 2017. In *Proceedings of the Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, pages 1–15, Valencia, Spain.