Natural Language Processing for Low-resourced Code-switched Colloquial Languages

The Case of Algerian Language

THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Natural Language Processing for Low-resourced Code-switched Colloquial Languages

The Case of Algerian Language

Wafia Adouane

Department of Philosophy, Linguistics and Theory of Science
Centre for Linguistic Theory and Studies in Probability
(CLASP)

Gothenburg, Sweden 2020

Doctoral dissertation in computational linguistics, University of Gothenburg

Cover: Thomas Ekholm
Printed by Repro Lorensberg,
University of Gothenburg
Gothenburg 2020
Publisher: University of Gothenburg (Dissertations)

ISBN 978-91-7833-958-7 (print)
ISBN 978-91-7833-959-4 (pdf)

Distribution

Department of Philosophy, Linguistics and Theory of Science
Box 200, SE-405 30 Gothenburg – Sweden

Abstract of the Thesis

In this thesis we explore to what extent deep neural networks (DNNs), trained end-to-end, can be used to perform natural language processing tasks for code-switched colloquial languages lacking both large automated data and processing tools, for instance tokenisers, morpho-syntactic and semantic parsers, etc. We opt for an end-to-end learning approach because this kind of data is hard to control due to its high orthographic and linguistic variability.

This variability makes it unrealistic to either find a dataset that exhaustively covers all the possible cases that could be used to devise processing tools or to build equivalent rule-based tools from the bottom up. Moreover, all our models are language-independent and do not require access to additional resources, hence we hope that they will be used with other languages or language varieties with similar settings.

We deal with the case of user-generated textual data written in Algerian language as naturally produced in social media. We experiment with five natural language processing tasks, namely Code-switch Detection, Semantic Textual Similarity, Spelling Normalisation and Correction, Sentiment Analysis, and Named Entity Recognition. For each task, we created a dataset from user-generated data reflecting the real use of the language.

Our experimental results in various setups indicate that end-to-end DNNs combined with character-level representation of the data are promising. Further experiments with advanced models, such as Transformer-based models, could lead to even better results. Completely solving the challenge of code-switched colloquial languages is beyond the scope of this experimental work. Even so, we believe that this work will extend the utility of DNNs trained end-to-end to low-resource settings. Furthermore, the results of our experiments can be used as a baseline for future research.

Acknowledgements

I am grateful to Jean-Philippe Bernardy for accepting to supervise this work with great patience and a lot of fun, and for his valuable guidance and useful feedback. I would like to sincerely thank very much Shalom Lappin, my co-supervisor, for all his support and inspiring discussions! I want also to thank Simon Dobnik for supervising this work for the first years. Special thanks go to all annotators for having contributed to this work in one way or another. I would like also to thank all my co-authors and colleagues at CLASP and FLoV for all the fruitful discussions we had. A special heartfelt thank you to my family for all.

The research reported in this thesis was supported by a grant from the Swedish Research Council (VR project 2014-39) for the establishment of the Centre for Linguistic Theory and Studies in Probability (CLASP) at the University of Gothenburg.

Wafia Adouane

Gothenburg – January 17th, 2020

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction to the Thesis

1 Background

Natural language processing (NLP) research has recently achieved outstanding results. In particular, utilising deep neural networks (DNNs) has pushed the field ahead, reaching ground-breaking performances for a wide range of tasks. Nevertheless, research is heavily focused on large, standardised, monolingual and well-edited corpora that exist only for a small set of well-resourced languages. More specifically, NLP research is still very English-centric (Schnoebelen, 2013) for whatever incentives (Hovy and Spruit, 2016) and domain-dependent —for instance, tools and models trained on well-edited large existing corpora for English (newswire or Wikipedia) have been shown to hardly work for social media texts written in English (Jørgensen et al., 2015). This kind of bias, present in much NLP research, has created serious issues of overgeneralisation and exclusion (Hovy and Spruit, 2016; Bender and Friedman, 2018) with factual direct or indirect social impact on people's daily life. For instance what kind of information they have access to or as simple as who to be friend with online and which video to watch next.

We believe that it is impractical to assume that all languages have so much linguistic similarity that they can be processed using the same methods and tools. Hence each language or language variety needs its own tools and models. Moreover, simply generalising existing NLP tools and models for English to all other languages is challenged by two facts. First, in real-world situations the majority of languages are low-resourced, i.e., they do not have ready-to-use data, let alone labelled data. Second, the unprecedentedly huge available data in new communication channels is unstructured. For our purposes, unstructured means that the generated data includes lots of colloquial languages which are unedited speech-like texts written in at least 2 languages or language varieties using spontaneous spelling. This situation occurs in particular in multilingual social environments, see user-generated examples: (4) a. in chapter 2 and (1) in chapter 3.

2 Research Question

The question that imposes itself is how to automatically process this kind of huge unstructured data to create tools and applications that can ease people's life, among others, automatic machine translation to enable people to have access to a wider divers content and more information, smart remote health care? From an NLP viewpoint and related to the scope of this work, the more precise question is how can we process user-generated textual data written in colloquial languages with no pre-existing NLP processing tools such as a tokeniser and a morpho-syntactic parser? Obviously, if achievable at all, it is extremely expensive and time consuming to develop such tools for every single language. A promising solution to avoid creating hand-crafting NLP processing tools for each language is simply to use the raw data with no processing or pre-processing. In other word, to what extent is it doable to replace rule-based and feature-based systems by end-to-end DNNs?

3 Contributions

In this experimental work, we attempt to answer this question by (1) gathering NLP resources for colloquial languages and (2) propose end-to-end DNNs capable of processing such resources. We work with the language used in Algeria (hereafter referred to as ALG) —we limit our scope to a national boundary because there are too many regional and local varieties— as a case study because it comprises all the linguistic and non-linguistic challenges mentioned above. Linguistically speaking, ALG is a mixture of languages and language varieties with heavy use of borrowings and code-switching, see user-generated examples: (4) a. in chapter 2, (1) in chapter 3 and (1) in chapter 4. Moreover, it is a colloquial language with high orthographic variability due to its lack of standardisation. Regrading non-linguistic challenges, although ALG is spoken by more than 42 million people [1], it is a low-resourced language in terms of NLP tools and applications.
Our main contributions could be summarised as follows.

1. We built, from scratch, 5 new benchmark corpora for ALG, manually labelled for the tasks of Code-switch Detection, Semantic Textual Similarity, a parallel corpus for Spelling Normalisation and Correction, Sentiment Analysis, and Named Entity Recognition. These corpora contain user-generated textual data reflecting the real-use of the language, and they are comparably the largest that exist for this language. We documented them following the data statement —a characterisation of a dataset that provides context— recommendations of Bender and Friedman (2018)

[1] https://en.wikipedia.org/wiki/Algeria

to make it easy for others to further explore this question or develop new NLP tools. This empirical data —naturally occurring language data— could be of use for documenting the language at hand and refining the existing theoretical frames, especially from a socio-linguistic perspective, even though this question is outside of the scope of this thesis.

2. We propose general end-to-end character-level models for each task along with exploring various ways to improve the performances, including bootstrapping, pre-trained language model, transfer learning, injecting background knowledge, data augmentation, and multi-task learning. The choice of an end-to-end deep learning approach is motivated by the high orthographic and linguistic variability of the data, making it unrealistic to either find a dataset that exhaustively covers all the possible cases that could be used to develop processing tools, or to build equivalent rule-based tools from bottom up. All our models are language-independent and do not require access to any additional resource, hence we hope that they will be used with other languages or language varieties in the same low-resource settings. Additionally, we provide a baseline for future research.

Our experimental results bring evidence towards a positive answer to the question earlier stated for various setups. Manifestly some setups work better than others for some tasks and classes, and other have comparable performances. Further experiments with advanced models could lead to even better results. We do not solve the problem, but we believe that this work will extend the utility of DNNs trained end-to-end to low-resource settings. To the best of our knowledge, the opportunity to explore end-to-end trained deep neural networks with code-switched colloquial Algerian language has not been previously realised.

A possible shortcoming of this work may reside in systematic biases which may exist in the collected data. To mitigate this issue, we situated our datasets for each task by describing their characteristics and the transformations we performed, if any, as much as possible. This way, any reader may be able to identify possible blend spots.

4 Ethical Considerations

As stated earlier we have built our linguistic corpora from online social media platforms because this data source suits the best our work, requiring spontaneously generated real-world data. This poses, nevertheless, a set of ethical concerns with regards to the nature of the collected data. In order to mitigate such concerns, from the beginning we sought to align our research with the ethical principles guiding information technology and computing, for instance the Association of Computing Machinery (Anderson, 1992), Menlo Re-

port (Dittrich and Kenneally, 2012), European Union General Data Protection Regulation (GDPR) (Council of European Union, 2016), as well as the European Union Regulations and the Ethics Assessment Proposal (Jansen et al., 2017).

Based on the definition of the Ethics Working Committee of the Association of Internet Researchers [2], data could be interactions, behaviours, transactions, production, presentation, performance, archived information, and locations and movements. Likewise, Article 4 of the GDPR [3] gives a broad definition of what is classified as personal data: '*personal data means any information relating to an identified or identifiable natural person*'. This includes: a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person, etc.

In the process of building our corpora, we followed the recommendations in the above mentioned resources aiming to protect human subjects against exploitation or what is referred to in Menlo Report as *the principles of respect for persons*. We first collected a list of online platforms where our language of interest is used. Informed consent is vital, but in our case there are too many people (often hundred of thousands and even millions for some platforms) which makes it unfeasible to contact and ask every user individually —in the author's lifetime. Instead we contacted the owners and the admins of the platforms and explained the purpose of our research project —to document the language at hand by creating NLP processing tools for it— and detailed our data processing. They were very cooperative and we got written permissions in shorter time than expected. They also published a copy of the permission publicly on their respective platforms, in case of any objection from the users. Many users contacted us and actually contributed to the data collection in the spirit of citizen science —we should call it citizen data though.

The final collected corpora include users generated texts without saving any meta information about the users themselves. We manually anonymised mentions of people included in the texts [4]. Likewise we comply with the GDPR on two lawful bases: (1) since our purpose is to carry out a public task with public interest (data is necessary to carry the task), we consider that we do not need to ask for explicit consent. Still we informed, at a general level, the users that the data will be used in our research. (2) We process only the right data (generated comments) with no other meta data because, as mentioned earlier, the goal is to analyse the language. We do not record any personal data (sensitive information) and individuals can not be identified based on the collected texts.

[2] http://aoir.org/reports/ethics.pdf

[3] https://eur-lex.europa.eu/legal-content/en/TXT/?uri=CELEX:32016R0679

[4] Randomly change proper names by others that are more general because we are interested in the sentence structure and the choice of the lexical items.

5 Structure of the Thesis

This thesis is based on the following papers published in peer-reviewed venues.

Paper 1 Wafia Adouane and Simon Dobnik. 2017. "Identification of Languages in Algerian Arabic Multilingual Documents". *In Proceedings of The 3rd Arabic Natural Language Processing Workshop (WANLP)*, pages 1–8. Association for Computational Linguistics. [Chapter 2]

Paper 2 Wafia Adouane, Simon Dobnik, Jean-Philippe Bernardy, and Nasredine Semmar. 2018. "A Comparison of Character Neural Language Model and Bootstrapping for Language Identification in Multilingual Noisy Texts". *In Proceedings of the 2nd Workshop on Subword and Character Level Models in NLP (SCLeM)*, pages 22–31. Association for Computational Linguistics. [Chapter 3]

Paper 3 Wafia Adouane, Jean-Philippe Bernardy, and Simon Dobnik. 2018. "Improving Neural Network Performance by Injecting Background Knowledge: Detecting Code-switching and Borrowing in Algerian texts". *In Proceedings of the 3rd Workshop on Computational Approaches to Linguistic Code-Switching*, pages 20–28. Association for Computational Linguistics. [Chapter 4]

Paper 4 Wafia Adouane, Jean-Philippe Bernardy, and Simon Dobnik. 2019. "Neural Models for Detecting Binary Semantic Textual Similarity for Algerian and MSA". *In Proceedings of the 4th Arabic Natural Language Processing Workshop (WANLP)*, pages 78–87. Association for Computational Linguistics. [Chapter 5]

Paper 5 Wafia Adouane, Jean-Philippe Bernardy, and Simon Dobnik. 2019. "Normalising Non-standardised Orthography in Algerian Code-switched User-generated Data". *In Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT)*, pages 131–140. Association for Computational Linguistics. [Chapter 6]

Paper 6 Wafia Adouane, Samia Touileb, and Jean-Philippe Bernardy. 2020. "Identifying Sentiments in Algerian Code-switched User-generated Comments". *In Proceedings of the 12th International Conference on Language Resources*

and Evaluation (LREC 2020), pages 2691–2698. European Language Resources Association. [Chapter 7]

Paper 7 Wafia Adouane and Jean-Philippe Bernardy. 2020. "When is Multi-task Learning Beneficial for Low-Resource Noisy User-generated Algerian Texts?" *In Proceedings of the 4th Workshop on Computational Approaches to Linguistic Code-Switching*, pages 17–25. European Language Resources Association. [Chapter 8]

The contents of corresponding chapters are a reproduction of the published papers with minor exceptions: the order of sections was sometimes modified for consistency, some terms were changed for consistency, and the format was changed for uniformity. Each paper remains self-contained, and can be read independently from the rest of the thesis.

Statement of personal contribution For each of the papers, I was the main contributor with regard to the formulation of the research questions, the methodology, the preparation of the data, the design of the experiments, the implementation of the models, the analyses of the results, and the writing of the initial drafts of the papers. The rest of the contributions were shared with the co-authors. The exceptions are: (1) Jean-Philippe Bernardy contributed considerably to the design and the implementation of the DNN architectures for Paper 2, Paper 3, Paper 5, and Paper 7. He also implemented the aligner described in Section 4.2 of Paper 5 and helped implementing the CNN model in Paper 6. (2) Samia Touileb contributed to the data labelling, lexicon creation, implemented the SVM model and was responsible for the initial draft of Section 2 in Paper 6.

Chapter 2

Identification of Languages in Algerian Arabic Multilingual Documents

Wafia Adouane and Simon Dobnik

Abstract

This paper presents a language identification system designed to detect the language of each word, in its context, in a multilingual documents as generated in social media by bilingual/multilingual communities. As a case study we take speakers of Algerian language. We frame the task as a sequence tagging problem and use supervised machine learning with standard methods like HMM and Ngram classification tagging. We also experiment with a lexicon-based method. Combining all the methods in a fall-back mechanism and introducing some linguistic rules, to deal with unseen tokens and ambiguous words, gives an overall accuracy of 93.14%. Finally, we introduce rules for language identification from sequences of recognised words.

1 Introduction

Most of the current Natural Language Processing (NLP) tools deal with one language, assuming that all documents are monolingual. Nevertheless, there are many cases where more than one language is used in the same document –a text segment of any length. The present study seeks to fill in some of the needs to accommodate multilingual (including bilingual) documents in NLP tools. The phenomenon of using more than one language is common in multilingual societies where the contact between different languages has resulted in various language (code) mixing like code-switching and borrowings. Code-switching is commonly defined as the use of two or more languages/language varieties

with fluency in one conversation, or in a sentence, or even in a single word. Whereas borrowing is used to refer to the altering of words from one language into another.

There is no clear-cut distinction between borrowings and code-switching, and scholars have different views and arguments. We based our work on Poplack and Meechan (1998) who consider borrowing as the adaptation of lexical items, with a phonological and morphological integration, from one language to another. Otherwise, it is a code-switching, at single lexical item, phrasal or clausal levels, either the lexical item/phrase/clause exists or not in the first language.[1] We will use "language mixing" as a general term to refer to both code-switching and borrowing.

We frame the task of identifying language mixing as a segmentation of a document/text into sequences of words belonging to one language, i.e. segment identification or chunking based on the language of each word. Since language shifts can occur frequently at each point of a document we base our work on the isolated word assumption as referred to by Singh and Gorla (2007) who consider that it is more realistic to assume that every word in a document can be in a different language rather than a long sequence of words being in the same language. However, we are also interested in identifying the boundaries of each language use, sequences of words belonging to the same language, which we address by adding rules for language chunking.

This paper focuses mainly on the detection of language mixing in Algerian Arabic texts, written in Arabic script, used in social media while its contribution is to provide a system that is able to detect the language of each word in its context. The paper is organised as follows. In Section 2 we survey some related work. In Section 3 we give a brief overview of Algerian Arabic which is a well suited, and less studied, language for detecting language mixing. In Section 4 we present our newly built linguistic resources, from scratch, and we motivate our choices regarding the labelling of the data. In Section 5 we describe the different methods used to build our system and discuss our results. In Section 6 we conclude with the main findings and outline some of our future directions.

2 Related Work

There is an increasing need to accommodate multilingual documents in different NLP tasks. Most work focuses on detecting different language pairs in multilingual texts, among others, Dutch-Turkish (Nguyen and Doğruöz, 2013), English-Bengali and English-Hindi (Das and Gambäck, 2013), English-French (Carpuat, 2014), Swahili-English (Piergallini et al., 2016). Since 2014, a Shared Task on Language Identification in Code-Switched Data is also organised (Solorio et al., 2014).

[1] Refers to the first language the speakers/users use as their mother tongue.

Detecting language mixing in Arabic social media texts has also attracted the attention of the research community. Elfardy et al. (2013) propose an automatic system to identify linguistic code switch points between MSA and dialectal Arabic (Egyptian). The authors use a morphological analyser to decide whether a word is in MSA or DA, and they compare the performance of the system to the previous one (Elfardy and Diab, 2012) where they used unsupervised approach based on lexicons, sound-change rules, and language models. There is also work on detecting language mixing in Moroccan Arabic (Samih and Maier, 2016). In contrast to the previous work on Arabic, our labelling scheme and the system make a distinction between code-switching and borrowing which they do not consider. We also detect words in their contexts and do not group them in a Mixed class. To the best of our knowledge, we are not aware of any similar system which identifies language mixing in Algerian Arabic documents.

3 Algerian Arabic

Algerian Arabic is a group of North African Arabic dialects mixed with different languages spoken in Algeria. The language contact between many languages, throughout the history of the region, has resulted in a rich complex language comprising words, expressions, and linguistic structures from various Arabic dialects, different Berber varieties, French, Italian, Spanish, Turkish as well as other Mediterranean Romance languages. Modern Algerian Arabic is typically a mixture of Algerian Arabic dialects, Berber varieties, French, Classical Arabic, Modern Standard Arabic, and a few other languages like English. As it is the case with all North African languages, Algerian Arabic is heavily influenced by French where code-switching and borrowing at different levels could be found.

Algerian Arabic is different from Modern Standard Arabic (MSA) mainly phonologically and morphologically. For instance, some sounds in MSA are not used in Algerian Arabic, namely the interdental fricatives 'ث' /θ/, 'ذ' /ð/ and the glottal fricative 'ه' /h/ at a word final position. Instead they are pronounced as aspirated stop 'ت' /t/, dental stop 'د' /d/ and bilabial glide 'و' /w/ respectively. Hence, the MSA word ذهب /*hb/ "gold" is pronounced/written as 'دهب' /dhb/ in Algerian Arabic. Souag (2000) gives a detailed description of the characteristics of Algerian Arabic and describes at length how it differs from MSA. Compared to the rest of Arabic varieties, Algerian Arabic is different in many aspects (vocabulary, pronunciation, syntax, etc.). Maybe the main common characteristics between them is the use on non-standard orthography where people write according to their pronunciation.

4 Corpus and Lexicons

In this section we describe how we collected and labelled our corpus and explain the motivation behind some labelling decisions. We then describe how we build lexicons for each language and provide some statistics about each lexicon.

4.1 Corpus

We automatically collected content from various social media platforms that we knew they use Algerian Arabic. We included texts of various topics, structures and lengths. In total, we collected 10,597 documents. On this corpus we ran an automatic language identifier which is trained to distinguish between the most popular Arabic varieties (Adouane et al., 2016).

Afterwards, we only consider the documents that were identified as Algerian Arabic which gives us 10,586 documents (215,843 tokens). Note that we use *token* to refer to lexical words, sounds and digits (excluding punctuation and emoticons) and *word* to refer only to lexical words. For robustness, we further pre-processed the data where we removed punctuation, emoticons and diacritics, and then we normalised it. In social media users do not use punctuation and diacritics/short vowels in a consistent way, even within the same text. We opt for such normalisation because we assume that such idiosyncratic variation will not affect language identification.

Based on our knowledge of Algerian Arabic and our goal to distinguish between borrowing and code-switching at a single lexical item, we decided to classify words into six languages: Algerian Arabic (ALG), modern standard Arabic (MSA), French (FRC), Berber (BER)[2], English (ENG) and Borrowings (BOR) which includes foreign words adapted to the Algerian Arabic morphology. Moreover, we grouped all Named Entities in one class (NER), sounds and interjections in another (SND). Our choice is motivated by the fact that these words are language independent. We also keep digits to keep the context of words and grouped them in a class called DIG.

In total, we have nine separate classes. First, three native speakers of Algerian Arabic labelled the first 1,000 documents (22,067 words) from the pre-processed corpus, following a set of labelling guidelines which takes into account the above-mentioned linguistic differences between Algerian Arabic and Modern Standard Arabic. To assess the quality of the data labelling, we computed the inter-annotator agreement using the Cohen's kappa coefficient (κ), a standard metric used to evaluate the quality of a set of labels in classification tasks by assessing the annotators' agreement (Carletta, 1996). The κ on the human labelled 1,000 documents is 89.27%, which can be qualitatively interpreted as

[2]Berber is an Afro-Asiatic language used in North Africa and which is not related to Arabic.

"really good".

Next, we implemented a tagger based on Hidden Markov Models (HMM) and the Viterbi algorithm, to find the best sequence of language tags over a sequence of words. The assumption is that the context of the surrounding words and their language tags will predict the language for the current word. We apply smoothing – we assign an equal low probability (estimated from the training data) for unseen words – during training to estimate the emission probability and compute the transmission probabilities. We trained the HMM tagger on the human labelled 1,000 documents. We divided the remaining corpus (unlabelled data) into 9 parts (each part from 1-8 includes 1,000 documents and the last part includes 1,586 documents). We first used the trained tagger to automatically label the first part, then manually checked/corrected the labelling. After that, we added the checked labelled part to the already existing training dataset and used that to label the following part. We performed the same bootstrapping process until we labelled all the parts.

The gradual bootstrapping labelling of new parts of the corpus helped us in two ways. First, it speeded up the labelling process which took five weeks for three human annotators to check and correct the labels in the entire corpus compiled so far. It would take them far longer if they started labelling without the help of the HMM tagger. Second, checking and correcting the labelling of the automatic tagger served us to analyse the errors the tagger was making. The final result is a large labelled corpus with a human labelling quality which is an essential element for learning useful language models. Table 2.1 shows statistics about the current labelled corpus.

Class	ALG	MSA	FRC	BOR	NER	ENG	BER	DIG	SND
#Tokens	118,942	82,114	6,045	4,025	2,283	254	99	1,394	687

Table 2.1 Statistics about the labelled corpus.

4.2 Lexicons

We asked two other Algerian Arabic native speakers to collect words for each included language from the web excluding the platforms used to build the above-described corpus. We cleaned the newly compiled word lists and kept only one occurrence for each word, and we removed all ambiguous words: words that occur in more than one language. Table 2.2 gives some statistics about the final lexicons which are lists of words that unambiguously occur in a given language, one word per line in a `.txt` file. Effectively, we see the role of dictionaries as stores for exceptions, while for ambiguous words

we work towards a disambiguation mechanism.

Class	ALG	MSA	FRC	BOR	NER	ENG	BER
#Types	42,788	94,167	3,206	2,751	1,945	157	21,789

Table 2.2 Statistics about the lexicons.

5 Experiments and Results

In this section, we describe the methods and the different experimental setups we used to build our language identification tool. We analyse and discuss the obtained results. We start identifying language at a word level and then we combine words to identify the language of sequences. We approach the language identification at the word level by taking into account the context of these words. We supplement the method with a lexicon lookup approach and manually constructed rules.

To evaluate the performance of the system, we divided the final human labelled dataset into two parts: the training dataset which contains 10,008 documents (215,832 tokens) and the evaluation dataset which contains 578 documents (10,107 tokens). None of the documents included in the evaluation dataset were used to compile the lexicons previously described in Section 4.2.

5.1 Identifying words

5.1.1 HMM Tagger

In Section 4.1 we describe an implementation of a tagger based on Hidden Markov Models (HMM) used as a helping tool to bootstrap data labelling. Now, having a labelled corpus we are interested in the performance of the tagger on our final fully labelled corpus which we discuss here. We train the HMM tagger on the training data and evaluate it on the evaluation data. Table 2.3 shows the performance of the tagger.

The overall accuracy of the tagger is 85.88%. This quite high performance gives an idea about how useful and helpful was the use of the HMM tagger to label the data before the human checking. The tagger also outperforms the majority baseline (#majority class / #total tokens) which is 55.10%. From Table 2.3 we see that the HMM tagger is good at identifying ALG and MSA words, given an F-score of 88.50% and 85.99% respectively.[3]

[3]We ignore the DIG and SND classes because we are interested in lexical words. As explained above, we

Class	Precision (%)	Recall (%)	F-score
ALG	87.10	89.96	88.50
BER	100	18.18	30.77
BOR	97.71	40.38	57.14
DIG	100	94.74	97.30
ENG	100	24.14	38.89
FRC	82.28	63.87	71.92
MSA	84.03	88.04	85.99
NER	84.07	61.69	71.16
SND	100	85.71	92.31

Table 2.3 Performance of the HMM tagger.

However, this performance dropped with other classes, it is even lower than the majority baseline for BER and ENG.

The confusion matrix of the tagger (omitted here due to space constraints) shows that all classes are confused either with ALG or MSA. This can be explained by the fact that ALG and MSA are the majority classes which means that both emission and transmission probabilities are biased to these two classes. The analysis of the most frequent errors shows that errors can be grouped into two types. The first type includes ambiguous words.

(1) a. الماتش مشري حارس خلا البيت يدخل

b. The match is bought, the goal keeper allowed the ball to enter.

In example (1), the word 'البيت' is "the goal" in French, the same word means "the house" in MSA and "the room" in ALG. Also the following word 'يدخل' which means " to enter" is used with all the possible meanings of 'البيت' (enter a house/ a room and ball enters). The second type of errors relates to unseen words in the training data. Because of the smoothing we used, the HMM tagger does not return 'unseen word'. Instead, another tag is assigned, mostly ALG and MSA. We could identify such words by setting manually some thresholds, but it is not clear what these should be.

The Precision is high for all unambiguous tokens, however the Recall is very low. To overcome the limitation of the HMM tagger in dealing with unseen words, we decided

kept them to keep the context of each word.

to explore other methods. Moreover, we want to reduce the uncertainty of our tagger deciding what is an unseen word.

We found it difficult to set any threshold that is not data-dependent. Therefore, we introduced a new class called unknown (UNK) which is inspired from *active learning* (Settles, 2009). We believe that this should be used in all automatic systems instead of returning a simple guess based on its training model.

5.1.2 Lexicon-based Tagger

We devised a simple algorithm that performs a lexicon look-up and returns for each word the language of the lexicon it appears in (note that lexicons contain only unambiguous words). For SND, we created a list of most common sounds like 'بفف' "pff", 'هه' "hh". For digits, we used the `isdigit` method built-in Python. In the case where a word does not appear in any lexicon, the unknown UNK class is returned. This method does not require training, but it requires good quality lexicons with a wide coverage. We evaluated the lexicon-based tagger on the same evaluation dataset and the results are shown in Table 2.4.

Class	Precision (%)	Recall (%)	F-score
ALG	97.39	81.55	88.77
BER	100	63.64	77.78
BOR	98.52	83.91	90.63
DIG	100	100	100
ENG	100	55.17	71.11
FRC	96.30	84.85	90.21
MSA	97.69	82.43	89.42
NER	97.46	74.68	84.56
SND	100	100	100

Table 2.4 Performance of the lexicon tagger.

The overall accuracy of the tagger is 81.98%. From comparing the results shown in Table 2.4 and Table 2.3, it is clear that the Recall has increased for all classes except for ALG and MSA. The reason is that now we have the UNK class where among the 10,107 tokens used for evaluation, 1,610 words are tagged as UNK instead of ALG or MSA. We examined the UNK words and found that these words do not exist in the lexicons. Either they are completely new words or they are different spellings of already covered words (which count as different words).

The confusion matrix of the lexicon-based tagger (omitted here) shows that the most frequent errors are between all classes and the UNK class. The tagger often confuses

between ALG/MSA and MSA/ALG. It also occasionally confuses between ALG/FRC and ALG/NER. These errors could be explained by the fact that the context of a word is ignored.

(2) a. حطولنا بلا بقلاوا بلا ميقطعوه حرنا كيفاش نكلوه

b. They served us (a dish of) Baklava without cutting it, we did not know how to eat it.

In example (2) the first "بلا" means "dish" in French and the second "بلا" means "without" in MSA.

(3) a. وجدنا كلشي بلقيس لي قالولنا عليه

b. We prepared everything according to the measures they (gave) told us.

In example (3) the word "بلقيس" means "with the measure" in ALG and it is a female name (NER).

Analysing the tagging errors indicates that using lexicon-based tagger is not effective in dealing with ambiguous words because it ignores the context of words, and as known, the context is the main means of ambiguity resolution.

5.1.3 N-gram Tagger

Our goal is to build a language tagger, at a word level, which takes into account the context of each word in order to be able to properly deal with ambiguous words. At the same time, we want it to be able to deal with unseen words. Ideally we want it to return UNK for each word it did not see before. This is because we want to analyse the words the tagger is not able to identify and appropriately update our dictionaries.

The Natural Language Toolkit (NLTK) n-gram PoS tagger (Steven et al., 2009) is well suited for further experimentation. First, the tagging principle is the same and the only difference is the set of tags. Secondly, the NLTK Ngram tagger offers the possibility of changing the context of a word up to trigrams as well as the possibility of combining taggers (unigram, bigram, trigram) with the back-off option. It is also possible to select a

single class, for example the most frequent tag or UNK, as a default tag in case all other options fail.

This combination of different taggers and the back-off option leads to the optimisation of the tagger performance. We start with the method involving most knowledge/context, if it fails we back off progressively to a simpler method. Table 2.5 summarises the results of different configurations. We train and evaluate on the same training and evaluation sets as before.

Tagger	**Accuracy (%)**
Unigram	74.89
Bigram	12.27
Trigram	07.97
BackOff(Trigram, Bigram, Unigram, ALG)	87.12
BackOff(Trigram, Bigram, Unigram, UNK)	74.95
Default (ALG)	52.12

Table 2.5 Performance of different n-gram tagger configurations.

The use of bigram and trigram taggers alone has a very little effect because of the data sparsety. It is unlikely to find the same word sequences (bigram, trigram) several times. However, chaining the taggers has a positive effect on the overall performance. Notice also that tagging words with the majority class ALG performs less than the majority baseline, 52.12% compared to 55.10%. In Table 2.6, we show the performance of the BackOff(Trigram, Bigram, Unigram, UNK) tagger in detail.

Class	**Precision (%)**	**Recall (%)**	**F-score**
ALG	96.17	75.27	84.44
BER	100	27.27	42.86
BOR	99.24	41.01	58.04
DIG	100	94.74	97.30
ENG	100	20.69	34.29
FRC	97.38	60.61	74.71
MSA	97.45	79.48	87.55
NER	94.69	69.48	80.15
SND	100	85.71	92.31

Table 2.6 Performance of the BackOff tagger.

Compared to the previous tagger, this tagger suffers mainly from the unseen words where 2,279 tokens were tagged as UNK. This could account for the low Recall ob-

tained for all classes. There is also some confusion between MSA/ALG, ALG/MSA and FRC/ALG.

5.1.4 Combining n-gram taggers and lexicons

The unknown words predicted by the BackOff(Trigram, Bigram, Unigram, UNK) tagger can be replaced with words from our dictionaries. First, we run the BackOff(Trigram, Bigram, Unigram, UNK), and then we run the lexicon-based tagger to catch some of the UNK tokens. Table 2.7 summarises the results.

Class	Precision (%)	Recall (%)	F-score
ALG	96.47	92.88	94.64
BER	100	81.82	90.00
BOR	99.28	86.44	92.41
DIG	100	100	100
ENG	100	90.91	95.24
FRC	98.95	88.08	93.20
MSA	98.42	93.64	95.97
NER	96.05	94.81	95.42
SND	100	100	100

Table 2.7 Performance of the tagger combining n-gram and lexicons.

Combining information from the training data and the lexicons increases the performance of the language tagging for all classes, giving an overall accuracy of 92.86%. Still there are errors that are mainly caused by unseen and ambiguous words. Based on the confusion matrix of this tagger (omitted here) the errors affect the same language pairs as before.

All language tags are missing words that are tagged as UNK words (in total 476 words). We found that these words are neither seen in the training data nor covered by any existing lexicons new words or different (even as spelling variants of the existing words). Keeping track of the unseen words, by assigning them the UNK tag, allows us to extend the lexicons to ensure a wider coverage.

To test how data-dependent is our system, we cross-validated it, and all the accuracies were close to the reported overall accuracy of the system, combining n-grams and lexicons, evaluated on the evaluation data.

5.1.5 Adding rules

We analysed the lexicons and manually extracted some features that would help us identify the language, for instance the starting and the final sequence of characters of a word.

The application of these rules improved the performance of the system, given an overall accuracy of 93.14%, by catching some unseen vocabulary (the number of UNK dropped to 446). As shown in Table 2.8, this hybrid tagger is still unable to deal with unseen words in addition to confusing some language pairs due to lexical ambiguity.

		Misclassified									
		ALG	BER	BOR	DIG	ENG	FRC	MSA	NER	SND	UNK
Correct	ALG	4,912	0	0	0	0	4	56	1	0	295
	BER	1	9	0	0	0	0	0	0	0	1
	BOR	1	0	280	0	0	5	1	0	0	30
	DIG	0	0	0	38	0	0	0	0	0	0
	ENG	1	0	0	0	10	0	0	0	0	0
	FRC	28	0	0	0	0	384	0	0	0	16
	MSA	134	0	1	0	0	1	3,612	5	0	101
	NER	6	0	2	0	0	0	8	135	0	3
	SND	0	0	0	0	0	0	0	0	7	0

Table 2.8 Confusion matrix of the hybrid tagger.

5.2 Identifying sequences of words

Now that we have a model that predicts the class of each token in a text, we added rules to label also non-linguistic words (punctuation (PUN) and emoticons (EMO)). This helps us to keep the original texts as produced by users as well as PUN and EMO be might be useful for other NLP tasks like sentiment and opinion analysis. Based on this extended labelling, we designed rules to identify the language of a specific segment of a text. The output of the system is a chunked text (regardless of its length) identifying language boundaries. It is up to the user how to chunk language independent classes, i.e. NER, DIG and SND, either separately or include them in larger segments based on a set of rules. For instance, example (4) a. is chunked as in example (4) c..

(4) a. واش نذير يا نااااس راني توجور روطاااار الرفاي نتاعي مينوضنيش مام نريقليه ماشي لا فوط نتاعي 😬😂

b. What should I do people, I am always late my alarm clock does not wake me up even I set it , it is not my fault.

c. ALG[واش نذير] ALG[يا نااااس] MSA[راني] ALG[توجور روطاااار] FRC[الرفاي] BOR[نتاعي مينوضنيش] FRC[مام] BOR[نريقليه] ALG[ماشي] ALG[لا فوط] FRC[نتاعي] ALG[😂😬] EMO[😂😬]

Chunking text segments based on the language is entirely based on the identification of the language of each word in the segment. One of the open questions is what to do when words tagged as UNK are encountered. We still do not have a good way to deal with this situation, so we leave them as separate chunks UNK. Extending the training dataset and the coverage of the current lexicons would help to solve the problem.

6 Conclusions and Future Work

We have presented a system for identifying the language at word and long sequence levels in multilingual documents in Algerian Arabic. We described the data and the different methods used to train the system that is able to identify language of words in their context between Algerian Arabic, Berber, English, French, Modern Standard Arabic and mixed languages (borrowings). The system achieves a very good performance, with an overall accuracy of 93.14% against a baseline of the majority class of 55.10%.

We discussed the limitations of the current system and gave insights on how to overcome them. The system is also able to identify language boundaries, i.e. sequence of tokens, including digits, sounds, punctuation and emoticons, belonging to the same language/class. Moreover, it performs also well in identifying Named Entities. Our system trained on a multilingual data from multiple domains handles several tasks, namely context sensitive language identification at a word level (borrowing or code-switching), language identification at long sequence level (chunking) and Named Entity recognition.

In the future, we plan to evaluate the automatic lexicon extension, as well as use the system in tasks such as error correction, Named Entity classification (Person, Location, Product, Company), topic identification, sentiment analysis and textual entailment. We are currently extending our corpus and labelling it with other linguistic information.

Chapter 3

A Comparison of Character Neural Language Model and Bootstrapping for Language Identification in Multilingual Noisy Texts

Wafia Adouane, Simon Dobnik, Jean-Philippe Bernardy, and Nasredine Semmar

Abstract

This paper seeks to examine the effect of including background knowledge in the form of character pre-trained neural language model (LM), and data bootstrapping to overcome the problem of unbalanced limited resources. As a test, we explore the task of language identification in mixed-language short non-edited texts with a low-resourced language, namely the case of Algerian Arabic for which both labelled and unlabelled data are limited. We compare the performance of two traditional machine learning methods and a deep neural networks (DNNs) model. The results show that overall DNNs perform better on labelled data for the majority classes and struggle with the minority ones. While the effect of the untokenised and unlabelled data encoded as LM differs for each class, bootstrapping, however, improves the performance of all systems and all classes. These methods are language independent and could be generalised to other low-resourced languages for which a small labelled data and a larger unlabelled data are available.

1 Introduction

Most Natural Language Processing (NLP) tools are generally designed to deal with monolingual texts with more or less standardised spelling. However, users in social media, especially in multilingual societies, generate multilingual non-edited material where at

least two languages or language varieties are used. This phenomenon is linguistically referred to as language (code) mixing where code-switching and borrowing, among others, are the most studied phenomena. Poplack and Meechan (1998) defined borrowing as a morphological or a phonological adaptation of a word from one language to another and code-switching as the use of a foreign word, as it is in its original language, to express something in another language. However, the literature does not make it clear whether the use of different script is counted as borrowing, or code-switching or something else. For instance, there is no linguistic well-motivated theory about how to classify languages written in other scripts, like French written in Arabic script which is frequently the case in North Africa. This theoretical gap could be explained by the fact that this fairly recent phenomenon has emerged with the widespread of the new technologies.

In this paper, we consider both code-switching and borrowing and refer to them collectively as language mixing. Our motivation in doing so is to offer to sociolinguists a linguistically informative tool to analyse and study the language contact behaviour in the included languages. The task of identifying languages in mixed-language texts is a useful pre-processing tool where sequences belonging to different languages/varieties are identified. They are then processed by further language/variety-specific tools and models. This task itself has neither been well studied for situations when many languages are mixed nor has it been explored as a main or an auxiliary task in multi-task learning (Section 4).

In this paper, we explore two avenues for improving the state of the art in variety identification for Algerian Arabic. First, we measure the ability of recurrent neural networks to identify language mixing using only a limited training corpus. Second, we explore to what extent adding background knowledge in the form of pre-trained character-based language model and bootstrapping can be effective in dealing with low-resourced languages in the domain of language identification in mixed-language texts for which neither large labelled nor unlabelled datasets exist.

The paper is organised as follows. In Section 2 we briefly review related work and situate our work. In Section 3 we describe the linguistic landscape in Algeria to better motivate our work. In Section 4 we give a brief overview of methods for levering learning from limited datasets. In Section 5 we describe the data. In Section 6 we present the architecture of our learning configurations which include both traditional approaches and deep neural networks and explain the training methods used on the labelled data, experiments and results. In Section 7 we experiment with these models when adding background knowledge and report the results. In Section 8 we conclude with our main findings and outline our future work.

2 Related Work

There has been interesting work in detecting code mixing for a couple of languages/language varieties, mostly using traditional sequence labelling algorithms like Conditional Random Field (CRF), Hidden Markov Model (HMM), linear kernel Support Vector Machines (SVMs) and a combination of different methods and linguistic resources, (Elfardy and Diab, 2012; Elfardy et al., 2013; Barman et al., 2014a,b; Diab et al., 2016; Samih and Maier, 2016; Adouane and Dobnik, 2017) to name a few.

Prior work that is most closely related to our work using neural networks and related languages, Samih et al. (2016) used supervised deep neural networks (LSTM) and a CRF classifier on the top of it to detect code-switching, using small datasets of tweets, between Egyptian Arabic and MSA and between Spanish and English using pre-trained word embeddings trained on larger datasets. However, in their labelling they combined ambiguous words, which are words that could be of either languages depending on the context, in one class called 'ambiguous' and ignored words from minority languages. Moreover, the system was evaluated on a dataset with no instances of neither 'ambiguous' nor 'mixed-language' words, basically distinguishing between MSA and Egyptian Arabic words in addition to Named Entities and other non-linguistic tokens like punctuation, etc.

Similar to our work, Kocmi and Bojar (2017) proposed a supervised bidirectional LSTM model. However, the data used to train the model was created by mixing edited texts, at a line level, in 131 languages written in different scripts to create a multilingual data, making it a very different task from the one investigated here. We use non-edited texts, a realistic data as generated by users reflecting the real use of the included languages which are all written in the same Arabic script. Our texts are shorter and the size of the dataset is smaller, therefore, our task is more challenging.

By comparison to our work, most of the literature focuses on detecting code-switching points in a text, either at a token level or at a phrase level or even beyond a sentence boundaries, we distinguish between borrowing and code-switching at a word level by assigning all borrowed words to a separate variety (BOR). Most importantly, our main focus is to investigate ways to inject extra knowledge to take advantage of the unlabelled data.

3 Linguistic Situation in Algeria

Linguistic landscape in Algeria consists of several languages which are used in different social and geographic contexts to different degrees (Adouane et al., 2016). These include local Arabic varieties (ALG), Modern Standard Arabic (MSA) which is the only standardised Arabic variety, Berber which is an Afro-Asiatic language different from Arabic and

widely spoken in North Africa, and other non-Arabic languages such as French, English, Spanish, Turkish, etc. A typical text consists of a mixture of these languages, and this mixture is often referred to, somewhat mistakenly as Algerian Arabic. In this paper, we use the term Algerian language to refer to a mixture of languages and language varieties spoken in Algeria, and the term Algerian variety (ALG) to refer to the local variety of Arabic, which is used alongside other languages such as, for example Berber (BER).

This work seeks to identify the language or language variety of each word within an Algerian language text. Algerian language is characterised by non-standardised spelling and spelling variations based on the phonetic transcription of many local variants. For instance, the Algerian user-generated sentence in (1) is a mixture of 3 languages (Arabic, French and Berber) and 2 Arabic varieties (MSA and ALG). For a better visual illustration, we colour each word in (1) d. by its language, in (1) b. we give the IPA transcription, and in (1) c. we give a human English translation. To illustrate the difficulty of the problem, we show in (1) e. the (incorrect) translation proposed by *Google translate* where words in black are additional words not appearing in the original sentence.

(1) a. سيلتوبلي حل الطاقة و سكر لباب موراك

b. [murÃ¦k Ã¦lbÃ¦b sekkÃ¦r wu Ã¦tʕaqÃ¦ ħÃ¦l siːltupliː]

c. Please open the window and close the door behind you

d. French Algerian Berber MSA Berber MSA Algerian

e. SELTOPLEY POWER SOLUTION AND SUGAR FOR MORAK PAPER

All the words in different languages are normally written in the Arabic script, which causes high degree of lexical ambiguity and therefore even if we had dictionaries (only available for MSA) it would be hard to disambiguate word senses this way. In (1), the ALG word حل *open* means *solution* in MSA, the Berber word الطاقة *window* which is adapted to the MSA morphology by adding the MSA definite article ال (case of borrowing) means *energy/capacity* in MSA. The Berber word سكر *close* means *sugar / sweeten / liquor / get drunk* in MSA.

Moreover, the rich morphology of Arabic is challenging because it is a fusional language where suffixes and other morphemes are added to the base word, and a single morpheme denotes multiple aspects and features. Algerian Arabic shares many linguistic features with MSA, but it differs from it mainly phonologically, morphologically and lexically. For instance, a verb in the first person singular in ALG is the same as the first person plural in MSA. The absence of a morphological/syntactic analyser for ALG makes it challenging to correctly analyse an ALG text mixed with other languages and varieties.

Except for MSA, Arabic varieties are neither well-documented nor well-studied, and they are classified as low-resourced languages. Furthermore, social media are the only source of written texts for Algerian Arabic. The work in NLP on Algerian Arabic and other Arabic varieties also suffers severely from the lack of labelled (and even unlabelled) data that would allow any kind of supervised training.

Another challenge is that we have to deal with all the complications present in social media domain, namely the use of short texts, spelling and word segmentation errors, etc. in addition to the non-standard orthography used in informal Arabic varieties.

We see the task of identification of the variety of each word in a text a necessary first step towards developing more sophisticated NLP tools for this Arabic variety which is itself a mixture of other languages and varieties.

4 Leveraging Limited Datasets

Deep learning has become the leading approach to solving linguistic tasks. However deep neural networks (DNNs) used in a supervised and unsupervised learning scenario usually require large datasets in order for the trained models to perform well. For example, Zhang et al. (2015) estimated that the size of the training dataset for character-level DNNs for text classification task should range from hundreds of thousands to several million of examples.

The limits imposed by the lack of labelled datasets have been countered by combining *structural learning* and *semi-supervised learning* (Ando and Zhang, 2005). Contrary to the supervised approach where a labelled dataset is used to train a model, in *structural learning*, the learner first learns underlying structures from either labelled or unlabelled data. If the model is trained on labelled data, it should be possible to reuse the knowledge encoded in the relations of the predictive features in this auxiliary task, if properly trained, to solve other related tasks. If the model is trained on unlabelled data, the model captures the underlying structures of words or characters in a language as a language model (LM), i.e., model the probabilistic distribution of words and characters of a text.

Such pre-trained LM should be useful for various supervised tasks assuming that linguistic structures are predictive of the labels used in these tasks. Approaches like this

are known as *transfer learning* or *multi-task learning* (MTL) and are classified as a semi-supervised approaches (with no bootstrapping) (Zhou et al., 2004). There is an increasing interest in evaluating different frameworks (Ando and Zhang, 2005; Pan and Yang, 2010) and comparing neural network models (Cho et al., 2014b; Yosinski et al., 2014).

Some studies have shown that MTL is useful for certain tasks (Sutton et al., 2007) while others reported that it is not always effective (Martínez Alonso and Plank, 2017).

Bootstrapping (Nigam et al., 2000) is a general and commonly used method of countering the limits of labelled datasets for learning. It is a semi-supervised method where a well-performing model is used to automatically label new data which is subsequently used as a training data for another model. This helps to enhance supervised learning. However, this is also not always effective. For example, Pierce and Cardie (2001) and Ando and Zhang (2005) showed that bootstrapping degraded the performance of some of their classifiers.

5 Datasets

We use two datasets: a small dataset labelled with language labels, and a larger dataset lacking such labels. In the following we describe each of them.

5.1 Labelled data

We use the human labelled corpus described by Adouane and Dobnik (2017) in which each word is tagged with one of the following labels: ALG (Algerian), BER (Berber), BOR (Borrowing), ENG (English), FRC (French), MSA (Modern Standard Arabic), NER (Named Entity), SND (interjections/sounds) and DIG (digits). The annotators have access to the full context for each word. To the best of our knowledge, this corpus is the only available labelled dataset for code-switching and borrowing in Algerian Arabic, written in Arabic script, and in fact also one of the very few available datasets for this particular language variety overall. Because of the limited labelled resources the corpus is small, containing only 10,590 samples (each sample is a short text, for example one post in a social media platform).

In total, the data contains 215,875 tokens distributed unbalancely as follows: 55.10% ALG (representing the majority class with 118,960 words), 38.04% MSA (82,121 words), 2.80% FRC (6,049 words), 1.87% BOR (4,044 words), 1.05% NER (2,283 words), 0.64% DIG (1,392 numbers), 0.32% SND (691 tokens), 0.10% ENG (236 words), and 0.04% BER (99 words).

5.2 Unlabelled data

Unfortunately, there is no existing user-generated unlabelled textual corpus for ALG. Therefore, we also collected, automatically and manually, new content from social media in Algerian Arabic which include social networking sites, blogs, microblogs, forums, community media sites and user reviews.[1]

The new raw corpus contains mainly short non-edited texts which require further processing before useful information can be extracted from them. We cleaned and pre-processed the corpus following the pre-processing and normalisation methods described by Adouane and Dobnik (2017). The data pre-processing and normalisation is based on applying certain linguistic rules, including:

1. Removal of non-linguistic words like punctuation and emoticons (indeed emoticons and inconsistent punctuation are abundant in social media texts.).

2. Reducing all adjacent repeated letters to maximum two occurrences of letters, based on the principle that MSA allows no more than two adjacent occurrences of the same letter.

3. Removal of diacritics representing short vowels, because these are rarely used.

4. Removal of duplicated instances of texts.

5. Removal of texts not mainly written in Arabic script.

6. Normalisation all remaining characters to the Arabic script. Indeed, some users use related scripts like Persian, Pashto or Urdu characters, either because of their keyboard layout or to express some sounds which do not exist in the Arabic alphabet, e.g. /p/, /v/ and /g/.

Additionally, we feed each document, as a whole, to a language identification system that distinguishes between the most popular Arabic varieties (Adouane et al., 2016) including MSA; Moroccan (MOR); Tunisian (TUN); Egyptian (EGY); Levantine (LEV); Iraqi (IRQ) and Gulf (GUF) Arabic. We retain only those predicted to be Algerian language, so that we can focus on language identification within Algerian Arabic, at the word level.

Table 3.1 gives some statistics about the labelled and unlabelled datasets. Texts refer to short texts from social media, words to linguistic words excluding punctuation and other tokens, and types to sets of words or unique words. We notice that 82.52% of the words

[1] We have a documented permission from the owners/users of the used social media platforms to use their textual contributions for research.

occur less than 10 times in both datasets. This is due to the high variation of spelling and misspellings which are common in these kinds of texts.

Dataset	#Texts	#Words	#Types
Labelled	10,590	213,792	57,054
Unlabelled	189,479	3,270,996	290,629

Table 3.1 Information about datasets.

6 Using Labelled Data

6.1 Systems and Models

We frame the task as a sequence labelling problem, namely to assign each word in a sequence the label of the language that the word has in that context. We use three different approaches: two existing sequence labelling systems – (i) an HHM-based sequence labeller (Adouane and Dobnik, 2017); (ii) a classification-based system with various back-off strategies from (Adouane and Dobnik, 2017) which previously performed best on this task, henceforth called the state-of-the-art system; and (iii) a new system using deep neural networks (DNNs).

6.1.1 HMM system

The HMM system is a classical probabilistic sequence labelling system based on Hidden Markov Model where the probability of a label is estimated based on the history of the observations, previous words and previous labels. In order to optimise the probabilities and find the best sequence of labels based on a sequence of words, the Viterbi algorithm is used. For words that have not been seen in the training data, an constant low probability computed from the training data is assigned.

6.1.2 State-of-the-art system

The best-so-far performing system for identifying language mixing in Algerian texts is described by Adouane and Dobnik (2017). The system is a classifier-based model that predicts the language or variety of each word in the input text with various back-off-strategies: trigram and bigram classification, lexicon lookup from fairly large manually compiled and curated lexicons, manually-defined rules capturing linguistic knowledge based on word affixes, word length and character combinations, and finally the most frequent class (unigram).

6.1.3 DNN model

Recurrent Neural Networks (RNNs) (Elman, 1990) have been used extensively in sequence prediction. The most popular RNN variants are the Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and the Gated Recurrent Unit (GRU) (Cho et al., 2014b).

Our neural networks consists of four layers: one embedding layer, two recurrent layers, and a dense layer with softmax activation. All our models are optimised using the *Adam* optimiser, built using the *Keras* library (Chollet, 2015), and run using a TensorFlow backend. A summary of the model architecture is shown in Figure 3.1. (This variant is composed of only the uncoloured (white) parts of the figure; the coloured parts are added in the model described in Section 7). The DNN is provided the input character by character. We opt for character-based input rather than word-based input for two reasons. First, we expect that the internal structure of words (phonemes and morphemes) is predictive of a particular variety. This way we hope to capture contexts within words and across words. Second, we do not have to worry about the size of the vocabulary, which we would if we were to use word embeddings.

This language-identification model is trained end-to-end. Because of the nature of RNNs, the network will assign one language variant per input symbol, and thus per character — even though the tags are logically associated word-by-word. To deal with this mismatch, when training we tag each character of a word and the space which follows it with the variant of the word. When evaluating the model, we use the tag associated with the space, so that all the word has been fed to the model before a prediction is made.

We have trained models with various values for the hyper-parameters: number of layers, number of epochs, memory size, drop-out rate and the batch size, but report detailed results for the model with the best behaviour. We experimented with both GRU and LSTM RNNs and found that the GRU performs better than LSTM on our task which is in line with the results of the previous comparisons but on different tasks (Chung et al., 2014). We also found out that our best systems are optimised with the architecture shown in Figure 3.1 with a memory size of 200, batch size of 512 and number of epochs of 25. Increasing or decreasing these values caused the overall accuracy to drop. Using drop-out improved the performance of the systems (overall accuracy $> 90\%$) over not using it ($< 70\%$). The best results are obtained using drop-out rate of 0.2 for the recurrent layers. We refer to this model as *DNN* in the following.

6.2 Results and discussions

To ensure a fair comparison, all the models have been evaluated under the same conditions. We use 10-fold cross validation on all of them and report their performance mea-

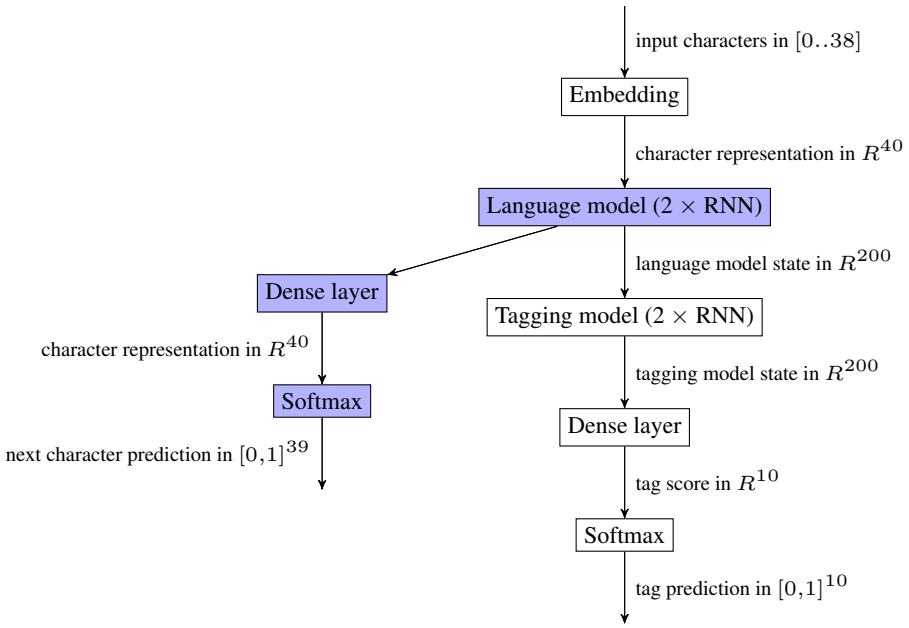


Figure 3.1 DNN architecture.

sured as the average accuracy. Table 3.2 shows the results. Note that for the DNN we only report the results of the (best-performing) GRU models. As a baseline we take the most frequent class in the labelled data. State-of-the-art (2) outperforms slightly HMM (1). DNN (3) outperforms slightly the State-of-the-art (2). All the systems perform better than the baseline.

	Model	Accuracy (%)
1	HMM	89.29
2	State-of-the-art	89.83
3	DNN	**90.53**
4	Baseline	55.10

Table 3.2 Performance of the models on labelled data.

Figure 3.2 shows the performance of each model per class reported as average F-score. Overall the models perform better on the majority classes such as ALG (Algerian) and MSA (Modern Standard Arabic), and non linguistic classes like DIG (digits) and SND

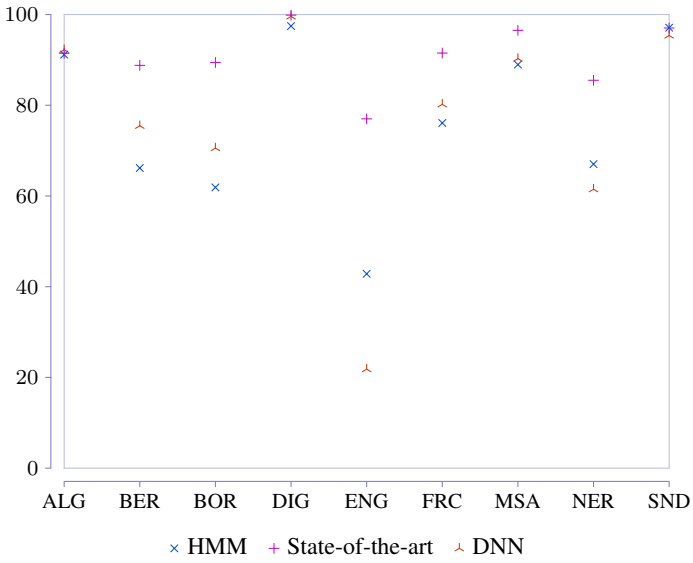


Figure 3.2 Models' average F-score per class.

(sounds) because their patterns are more or less regular and language independent. The State-of-the-art system achieves the best performances for all classes except for ALG where it is slightly outperformed by DNN, average F-score of 91.45 and 92.22 respectively. A possible explanation for this is that the State-of-the-art system is more robust because it involves several strategies of classification. DNN performed better than HMM in all cases except for ENG (English) and SND. Both DNN and HMM struggle with minority classe like ENG, BOR (borrowing), BER (Berber), NER (Named Entities), and FRC (French). Note that in this experiment we only used the smaller labelled dataset. In the following section, we explore ways to take advantage of the additional relatively large unlabelled dataset in order to improve the performance of the systems.

7 Using Data Augmentation with Background Knowledge

7.1 Training Methods

In this section, we examine which data augmentation method performed on the unlabelled corpus can best enhance the performance of our three models. We experiment with data bootstrapping, pre-training a language model, and the combination of both methods. In each case, we are providing some form of background knowledge compared to the task described in Section 6.

7.1.1 Bootstrapping

For bootstrapping, we use the State-of-the-art system (Section 6.1.2) to label the unlabelled data without additional checking of the quality of labels and then use this bootstrapped data in further training.

We re-run the experiments described in Section 6 using the bootstrapped data as the training data. We refer to the systems as *HMM bootstrapped*, *State-of-the-art bootstrapped*, and *DNN bootstrapped* respectively.

7.1.2 Language Model

Another way to take advantage of the unlabelled data is to train a language model (LM) on the whole data and use the internal state of the LM as input to the tagger, rather than using the raw textual input. To this end, we modify the structure of our DNN as indicated by the blue-coloured parts in Figure 3.1. Namely, we add two language-modelling RNN layers between the embedding and the tagging layers. They are followed by a dense layer with softmax activation, which predicts the next character in the input.

With this setup, we train the language-modelling layers on the unlabelled corpus, as a generative language model on the unlabelled data set. Thus, the output of these layers contains the information necessary to predict the next character given the previous sequence of characters. The language model is trained on 80% of the unlabelled data and evaluated on the remaining 20%. The rest of the network is then trained as in the previous case (Section 6.1.3). We stress that, in this instance, only the last two layers are trained on the language-identification task. We refer to this model as *DNN with LM*.

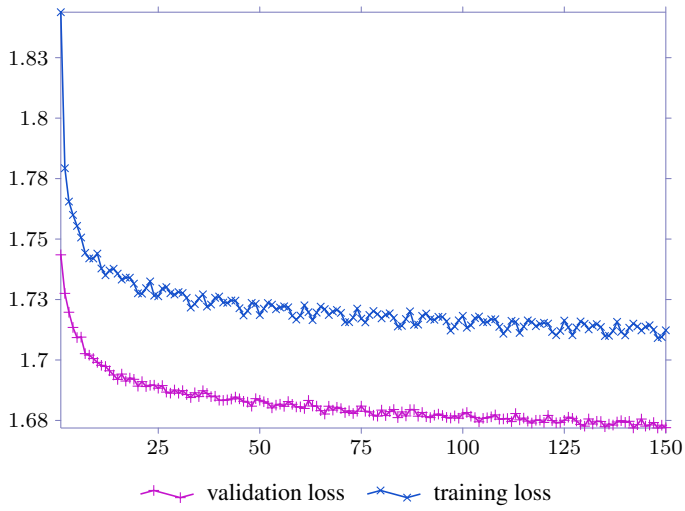


Figure 3.3 Language model loss through training epochs.

You may notice in Figure 3.3 that the model is still improving (at 150 epochs), albeit slowly, even after exhausting our computational budget. Nevertheless, the model appears to be working well as a text generator. For instance, we took sentence in (1) as a seed and obtained sentences that are grammatically and structurally acceptable, even if they are semantically meaningless and reproduce the many spelling variants found in the original corpus. Here are two examples:

1. المهم انا ما نقدرش نموت عليها و الله يا ختي راني معاك و الله ما نقولك انا ما نقدرش نتعلم من الحال
2. و الله ما نقول الله يبارك ما يعرفش يتبلاو و يعرف وين راه المرا الي ما يحبش يحب يقول الله يهدينا

7.1.3 Language Model and Bootstrapping

We retrain the DNN model using the pre-trained LM and the bootstrapped data in order to optimise the use of the unlabelled data. We refer to this model as *DNN bootstrapped and LM*.

7.2 Results and discussions

We evaluate all the models under the same conditions as in Section 6, using 10-fold cross validation we report the average accuracy over the folds.

The evaluation set in the bootstrapping models in each fold is only taken from the labelled data while the training part consists of a combined 9-folds from the labelled data and the entire bootstrapped data. In other words, the entire bootstrapped data is added to the training data at each time. In the case of DNNs, we found again that GRUs perform significantly better than LSTM, and that bootstrapped models are optimised with drop-out rate of 0.2 whereas models with language model perform better with drop-out rate of 0.1. The obtained results are reported as the average accuracy in

Table 3.3. For the DNN, we only report the results of the (best-performing) GRU models.

	Model	Accuracy (%)
1	HMM bootstrapped	93.97
2	State-of-the-art bootstrapped	**95.42**
3	DNN bootstrapped	93.31
4	DNN with LM	90.31
5	DNN bootstrapped and LM	90.19

Table 3.3 Performance of the models with background knowledge.

The best performance overall is achieved by the bootstrapped state-of-of-the-art model (2). HMM bootstrapped (1) performs slightly better than the DNN bootstrapped (3). Bootstrapping helps the State-of-the-art system and HMM more than DNN. This is due to the training nature of the DNN which is based on capturing frequent regular patterns, hence adding the bootstrapped data means introducing even more irregular patterns.

Compared to the results in Section 6.2, the DNN bootstrapped (3) outperforms all the models with the labelled data: (1), (2) and (3) in Table 3.2. The bootstrapping method thus improves the performance of all configurations, whether they are using DNNs or not. The reported benefits of bootstrapping are contrary to the previous observations where bootstrapping did not help (Section 4).

However, the use of the language model (4) decreases slightly (-0.22%) the performance of the DNN compared to its performance with the labelled data (3) in Table 3.2. The use of the bootstrapping and the language model (5) leads to no significant difference in performance in respect to (4). Overall, it appears that the usage of the language model has no strong effect. This could be caused by the noise in the data, and adding more unlabelled data makes it hard for the language model to learn all the data irregularities. Maybe the system requires more training data.

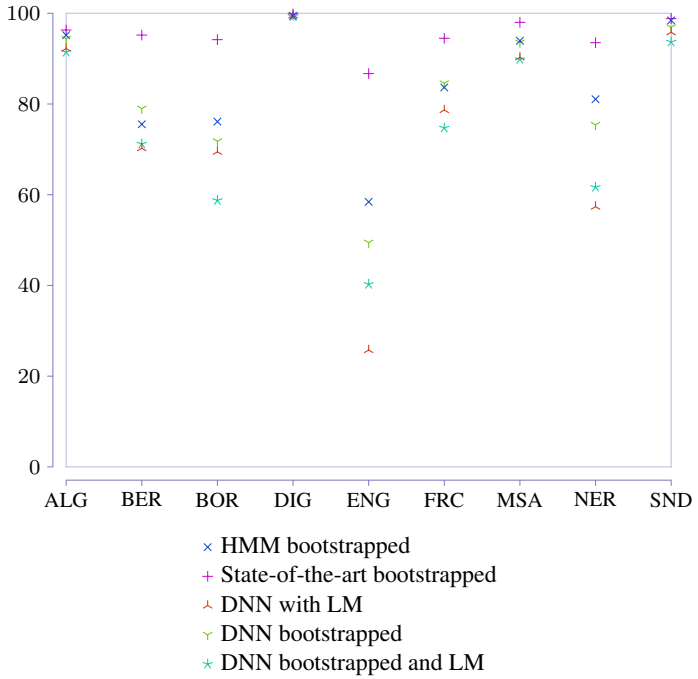


Figure 3.4 Models' average F-score per class.

Figure 3.4 sums up the performance of each model per class reported as the average F-score. The first thing to notice is that bootstrapping improves the performance of all systems, and the best performance is achieved with the State-of-the-art. This could be explained by 'the more data, the better performance'. HMM bootstrapped outperforms the DNN bootstrapped except for FRC and BER. Adding language model to the DNN causes the overall accuracy to drop compared to the DNN bootstrapped. Nevertheless, compared to the results in Figure 3.2, language model behaves differently with each class. For instance, it boosts the performance of the DNN on ENG, and the performance on BOR, BER, FRC over HMM. Whereas combining language model and bootstrapped data performs the worst except for BER, ENG and NER. The effect of combining bootstrapping and language model is better for minority classes: BER, ENG and NER.

Error analysis of the confusion matrices shows that all the systems are confused, chiefly between ALG and MSA, BOR and ALG, FRC and ALG. The confusions are caused mainly by the lexical ambiguity between these classes, given that we identify the language of each word in its context.

8 Conclusions and Future Work

We have examined the automatic classification of language identification in mixed-language texts on limited datasets of Algerian Arabic, in particular a small unbalanced labelled dataset and a slightly larger unlabelled dataset. We tested whether the inclusion of a pre-trained LM on the unlabelled dataset and bootstrapping the unlabelled dataset can leverage the performance of the systems. Overall when using only the small labelled data, DNNs outperformed the HMM and the State-of-the-art system. However, DNNs performed better on the majority classes and struggled with the minority ones in comparison to the State-of-the-art system. Bootstrapping improved the performance of all models, both DNNs and not DNNs for all classes.

Adding a background knowledge in the form of a pre-trained LM to DNNs had a different effect per class. While it boosted the performance of the minority classes, its effect on the majority ones was not clear. Despite the generative behaviour of the LM, tested in Section 7.1.2, which showed that LM did learn the underlying structures of the unlabelled data, the effect of the encoded knowledge maybe was not suitable for our main task. This could be also caused by the high noise level in the data, even though deep learning is generally thought to handle noise well.

In our future work, we will focus on exploring (i) different DNN configurations to investigate the best ways of injecting background knowledge as well as (ii) different data pre-processing methods to normalise spelling and remove misspellings for MSA, and deal with word segmentation errors.

Chapter 4

Improving Neural Network Performance by Injecting Background Knowledge: Detecting Code-switching and Borrowing in Algerian Texts

Wafia Adouane, Jean-Philippe Bernardy, and Simon Dobnik

Abstract

We explore the effect of injecting background knowledge to different deep neural network (DNN) configurations in order to mitigate the problem of the scarcity of labelled data when applying these models on datasets of low-resourced languages. The background knowledge is encoded in the form of lexicons and pre-trained sub-word embeddings. The DNN models are evaluated on the task of detecting code-switching and borrowing points in non-standardised user-generated Algerian texts. Overall results show that DNNs benefit from adding background knowledge. However, the gain varies between models and classes. The proposed DNN architectures are generic and could be applied to other low-resourced languages.

1 Introduction

Recent success of DNNs in various natural language processing (NLP) tasks has attracted attention from the research community attempting to extend their application to new tasks. Nevertheless, the large amount of labelled data required to train DNNs limits their application to new tasks and new languages because it is hard to find large labelled corpora for these domains. The issue is even more severe for low-resourced languages.

Another serious problem with most current NLP approaches and systems is that they are trained on well-edited standardised monolingual corpora, such as the Wall Street Journal, Wikipedia, etc. This could be explained by the fact that for a long time NLP has been influenced by the dominant descriptive linguistic theories affected by the standard

language ideology which assumes that natural languages are uniform and monolingual. However, standardisation is not universal as stated by Milroy (2001), meaning that not all languages are standardised. Therefore, lexical, structural and phonological variation is, for instance, the norm in natural language and not an exception, meaning that well-edited texts do not really reflect the natural usage of natural languages, but only represent formal languages.

The discrepancy between the assumed uniformity of language both in linguistic theory and NLP and their variable nature is accentuated by new technologies, such as social media platforms and messaging services. These new communication platforms have facilitated the proliferation of writing in non-standardised languages on the web, such as colloquial Arabic or what is commonly referred to as dialectal Arabic. This is because in interactive scenarios people usually use spoken-like (colloquial) language or, in multilingual societies where people have access to several linguistic codes at the same time, a mixture of languages/language varieties. Consequently, this new kind of written data has created a serious problem regarding the usability of the existing NLP tools and approaches as they fail to properly process it, even in the case of well-resourced languages.

The contribution of the paper is to explore how to mitigate the problems (i) of the scarcity of labelled data when using DNNs with low-resourced languages, and to what extent can we take advantage of the limited available resources, and (ii) to provide NLP approaches and tools that would be able to deal with non-standardised texts and language-mixing. In particular, for (i) we investigate what are the optimal ways of injecting available background knowledge to different configurations of DNNs in order to improve their performance. For (ii) we take the case of the language used in Algeria as it poses serious challenges for the available NLP approaches and tools. It is a low-resourced multilingual colloquial language. We chose the task of a word-level language identification which is a first step towards processing such texts. The task focuses on detecting code-switching and borrowing points in a text which represents the same utterance. Knowing what parts of text belong to what language variety allows to perform better qualitative and quantitative analysis of such texts with other tools.

The paper is organised as follows. In Section 2 we compare our contribution to previous related work. In Section 3 we briefly describe the complex linguistic situation in Algeria as a result of a language contact. The section aims to explain the linguistic challenges of processing such texts and motivates our choices based on established sociolinguistic theories. In Section 4 we present our available linguistic resources. In Section 5 we describe our different DNN configurations. In Section 6 we describe our experimental setup and analyse the results. Finally, in Section 7 we conclude with the main findings and outline some of our future plans.

2 Related Work

The emerging digitised multilingual data that followed the introduction of new technologies and communication services has attracted attention of the NLP research community in terms of how to process such linguistic data that resulted from language contact between several related and unrelated languages, for example in detection of code-switching where mainly traditional sequence labelling methods are used for, among others, Bengali-English-Hindi (Barman et al., 2014a), Nepali-English (Barman et al., 2014b), Spanish-English and MSA-Egyptian Arabic (Diab et al., 2016), MSA-Moroccan Arabic (Samih and Maier, 2016), MSA-Algerian Arabic-Berber-French-English (Adouane and Dobnik, 2017), etc.

The work most closely related to ours is by Samih et al. (2016) who used a supervised LSTM-RNN model combined with Conditional Random Fields to detect switching points between related languages (MSA - Egyptian Arabic) trained on a small dataset from Twitter. However, the system was only evaluated on the majority classes. Similarly, Kocmi and Bojar (2017) proposed a supervised bidirectional LSTM-RNN trained on artificially created multilingual edited texts. These does not fully reflect all the complexities of real linguistic use in a multilingual scenario.

Adouane et al. (2018b) propose a character-level GRU-RNN on the same task as described here backed by the available unlabelled data. They report that their supervised RNN model performs the best on labels with more representative samples. Adding neural language model that was pre-trained on noisy unlabelled data does not help, but bootstrapping the unlabelled data with another system improves the performance of all their systems.

In this work we use different DNN architectures (RNNs and CNNs), and we aim to examine the behaviour of each model when injecting background knowledge in the form of encoded information from the available lexicons and a pre-trained sub-word embeddings from unlabelled data. Our goal is to take advantage of the available NLP resources, with as little processing as possible to mitigate the problem of scarce labelled data.

3 Linguistic Background

In North Africa in general, and in Algeria in particular, intense language contact between various related and unrelated languages has resulted in a complex linguistic situation where several languages are used in a single communicative event. A few cases of language contact have attracted the attention of the linguistic community while the monolingual norm dominates in linguistics. One kind of language contact situation has been described by Ferguson (1959) as *diglossia* which refers to a situation where two linguistic

systems coexist in a functional distribution within the same speech community.

In another kind of language contact situations, several languages coexist but not in a well-defined functional distribution. This situation is referred to as *bilingualism* (Sayahi, 2014) which could result from either informal contact between coexisting languages like Berber and Arabic, or from formal education where in addition to other language people learn French with varying degrees of competence.

Based on the Fishman's model (Fishman, 1967), North African Arabic, known as Maghrebi Arabic, is classified as a linguistic situation in the speech community characterised by diglossia with bilingualism. The intense language contact between related and unrelated languages has resulted mainly in two widespread linguistic phenomena: code-switching and borrowing. As defined by Poplack and Meechan (1998), code-switching is (ideally) integration of material from one language to another without any phonological, morphological or syntactic integration, whereas borrowing is when material is integrated.

For computational purposes, we focus on *diglossic code-switching* (Sayahi, 2014), which happens between related languages such as switching between Arabic varieties, and *bilingual code-switching*, which happens between unrelated languages such as switching between one Arabic variety and other coexisting language such as Berber, French or English. Regarding borrowing, it is practically not possible to clearly distinguish whether a word in one Arabic variety is integrated into another variety or not because there are no lexicons for Arabic varieties, except for the standard one, and we also do not have access to acoustic representations of words. Based on this, we can practically focus only on *bilingual borrowing* rather than on *diglossic borrowing*.

(1) a. ديري سربيتة صغيرا في طاسة تاع ما او دوبي فيها اسبيجيك وكمديلو بيها نوغمال هاك مداري ندير يريح تم تم

b. Put a small towel in a cup of water and dissolve Aspegic in it and cover him with it, it is what I usually do. He will feel quickly better.

As illustration, the example in (1) is a user-generated utterance which contains words in Modern Standard Arabic (صغيرا، في، فيها), note that the word صغيرا is misspelled and should be spelt like صغيرة, words in local Algerian Arabic (ديري، تاع، ما، او، دوبي، وكمديلو، بيها، هاك، مداري، ندير، يريح، تم تم), French words integrated in Arabic (سربيتة، طاسة) , and a

French word without integration (نوغمال).

4 Linguistic Resources

We use the dataset by Adouane and Dobnik (2017) where each word is tagged with a label identifying its class which could be a language/language variety, including local Arabic varieties (ALG), Modern Standard Arabic (MSA), French (FRC), Berber (BER), English (ENG), non-Arabic words integrated in local Arabic or what is referred to as borrowing (BOR), in addition to language independent classes such as named entities (NER), digits (DIG) and interjections (SND). To the best of our knowledge this is the only available labelled dataset for code-switching and borrowing for Algerian. As the labelled dataset is small, we also collected a larger unlabelled dataset from the same sources as the authors of the labelled dataset, and pre-processed them in the same way. Table 4.1 gives information about the datasets where texts refer to social media texts with an average length of 19 words, words refer to linguistic words excluding other tokens (digits, punctuation, emoticons), and types refer to unique words.

Dataset	#Texts	#Words	#Types
Labelled	10,590	213,792	57,054
Unlabelled	311,130	4,928,827	350,759

Table 4.1 Statistics about the datasets.

We also use the lexicons compiled by the authors of the labelled dataset, with further cleaning. The lexicons include lists of inflected words checked manually, one list per class. Words belonging to more than one class are not included. Table 4.2 gives more information about the sizes of the lexicons.

Class	ALG	MSA	FRC	BOR	NER	ENG	BER
#Types	42,788	94,167	3,206	3,509	1,945	165	21,789

Table 4.2 Statistics about the lexicons.

5 Models

We approach the task of detecting code-switching and borrowing points in text as a sequence tagging problem where the aim is to assign a tag to each word in the text depending on its context. We use two DNN architectures, namely Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) with different configurations summarised in Figure 4.1.

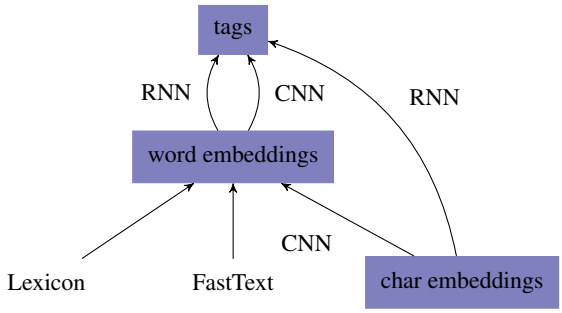


Figure 4.1 A summary of possible tagging models.

The first option is to use an RNN to map character embeddings to tags directly. Alternatively, we can use word embeddings. Word embedding can be any combination of (a) fixed lexicon information (b) fasttext embeddings (c) a custom CNN built from character embeddings. The word embeddings can be mapped to tags using either an RNN or a CNN, or a simple dense layer with softmax activation.

Except for the pure Lexicon-based model, all other models have access to characters and thus to the internal structure of words (phoneme and morphemes), which we expect to be predictive of a particular variety. All models are trained end-to-end, except for the fasttext embeddings and the lexicon. We report only the configurations of models which give the best performance, with the fine-tuned parameters, namely the number of units for each RNN layer, dropout rate, the number of features and the filter size for each CNN layer. The parameters are fine-tuned on a separate development set containing 1,000 texts (13,771 tokens).

5.1 Character-level RNN

The character-level RNN is composed of two LSTM layers of 400 units each, with a dropout of 10%, followed by a dense layer with softmax activation. Due to the nature of RNNs, the network assigns one language variant per input symbol, and thus per character

—but the task is to predict a tag for each word. To deal with this limitation, we consider only the tag associated with the last character of a word.

5.2 Word-level RNN

The word-level RNN is composed of a standard LSTM layer with 400 units with a dropout of 10%, followed by a dense layer.

5.3 Character-level CNN

The character-level CNN is composed of two convolution layers with 60 features with a filter size 5, with a ReLU activation and a dropout of 10%, followed by max pooling in the temporal dimension.

5.4 Word-level CNN

The word-level CNN is composed of two convolution layers with a filter size 3, with a ReLU activation and a dropout of 10%, followed by a dense layer with softmax activation. The first layer uses 100 features and the second 60 features.

5.5 Lexicon-based Model

In order to take advantage of the available lexicons, Table 4.2, we represent their words as one-hot encoding vector, which we refer to as lexicon embeddings. The lexicon-based model is composed of the lexicon embeddings followed by two convolution layers with a filter size 3, with a ReLU activation and a dropout of 10%, followed by a dense layer with softmax activation. The first layer uses 100 features and the second 60 features.

5.6 FastText-based Model

In order to take advantage of the unlabelled dataset, Table 4.1, containing a high level of misspellings and spelling variation, we assume that word embeddings that are based on sub-word information capture spelling variation and morphological information better than the embeddings that take word as a unit. For this purpose we use FastText library designed to train word embeddings where a word is represented as the sum of its sub-strings (Bojanowski et al., 2016). We created five fasttext embeddings trained on the unlabelled dataset with different parameters. We found that the optimal parameters are: word vector dimension of 300, and the range of the size of the sub-strings representing a word between 3 and 6 characters, with a context size of 5 words, trained on 20 epochs. The FastText-based model is composed of the fasttext embeddings followed by two convolution layers

with filter size 3, with a ReLU activation and a dropout of 10%, followed by a dense layer with softmax activation. The first layer uses 100 features and the second 60 features.

6 Experiments and Results

All models and configurations are evaluated under the same conditions using 10-fold cross-validation on the labelled dataset. As a baseline we take an existing system (Adouane and Dobnik, 2017), a classification-based system which uses a chain of additional back-off strategies which involve lexicons, linguistic rules, and finally the selection of the most frequent class. We refer to this system as the baseline.

First, we train the RNN and CNN models only on the labelled data (supervised learning) without any background knowledge. We also examine the effect of the FastText-based and the Lexicon-based models separately to quantify the contribution of each. Then we combine both models to optimise their performance. Second, in order to take advantage of all available linguistic resources, we add to each of the RNN and the CNN models background knowledge in the form of (i) lexicon embeddings; (ii) fasttext embeddings; (iii) a combination of both lexicon and fasttext embeddings; and (iv) bootstrap the unlabelled dataset with the baseline system and train the best performing DNN model on it to investigate whether bootstrapping improves its performance.

All results are reported as the average performance of the 10-fold cross-validation for each model at epoch 100 using the parameters mentioned earlier. For short, we use FastText to refer to the FastText-based model and fasttext to refer to the fasttext embeddings, Lexicon to refer to the Lexicon-based model and lexicon to refer to the lexicon embeddings.

6.1 Models without Background Knowledge

In Table 4.3 we report the average error rate of the experiments without background knowledge for only the best performing RNN, CNN, Lexicon, and FastText models.

	Model	Error Rate (%)
1	Char-level RNN	13.38
2	Char-level CNN	**8.18**
3	FastText	16.46
4	Lexicon	20.62
5	FastText + Lexicon	9.21
6	Baseline	9.52

Table 4.3 Average error rate of the models without background knowledge.

Results show that the baseline (6) outperforms the Char-level RNN (1), FastText (3) and Lexicon (4) models. However, the baseline is outperformed by the Char-level CNN model (2) with 1.34% error reduction. Combining FastText and Lexicon in one model (5) performs much better than using each model separately, and slightly outperforms the baseline by 0.31% error reduction.

In Figure 4.2 we report the average performance of each model per class, measured as Precision, Recall, F-score and Loss. Notice that we do not report the Loss for the baseline because of the way the system was designed. The results show that the baseline system performs better on the majority classes, ALG and MSA, with an average F-score of 91.91 and 90.44 respectively as well as on non-linguistic classes like DIG and SND with an average F-score of 97.17 and 93.88 respectively.

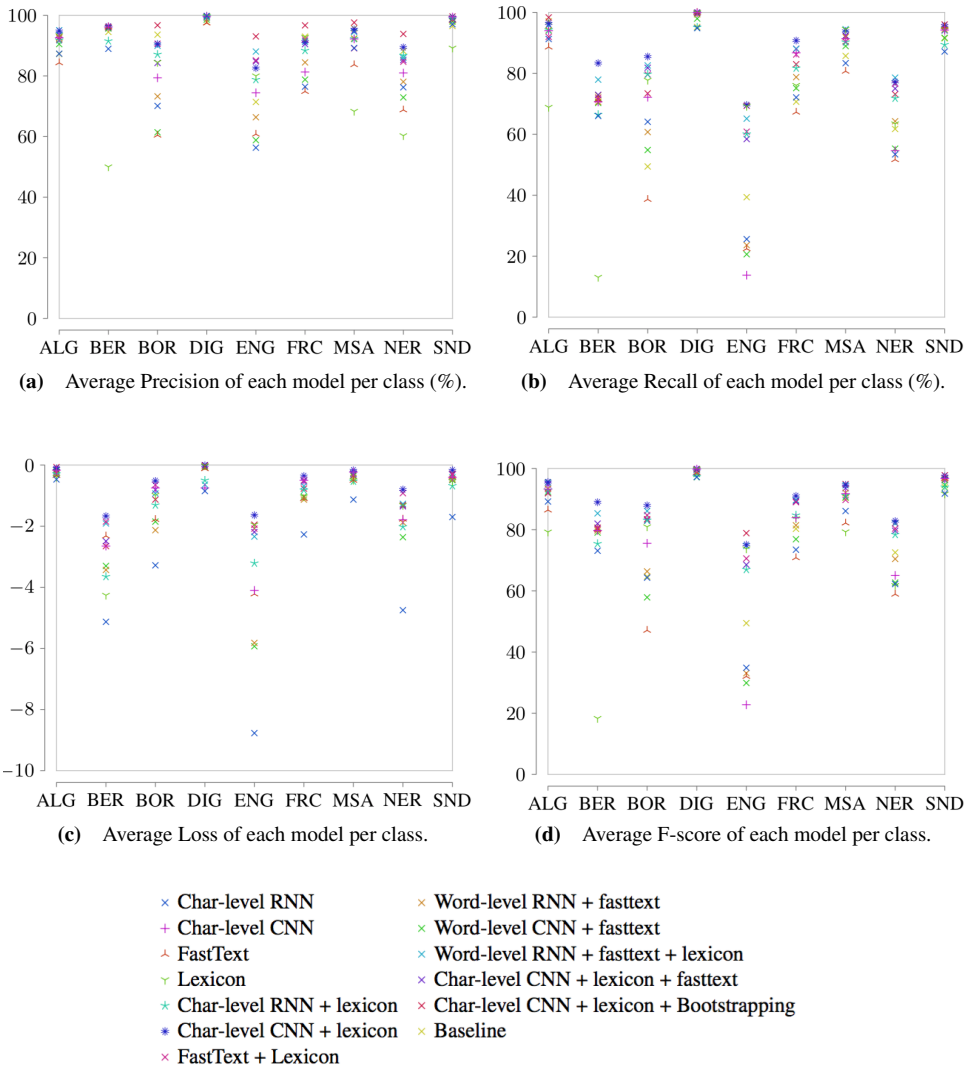
However, the baseline system performs less well on the minority classes, BER and FRC with an average F-score of 80.41 and 80.31 respectively, and performs even worse on NER and BOR with an average F-score of 72.55 and 64.70 respectively. It performs the worst on ENG with an average F-score of 49.45. Regarding the minority classes, Precision is high on BER (94.51%), BOR (93.61%), FRC (92.97%) and lower on NER (88.20%) and ENG (71.41%). However the Recall is low on all classes BER (72.76%), FRC (70.70%), NER (61.74%) and the lowest on BOR (49.44%), and ENG (39.37%).

The error analysis of the baseline system shows that the system is mostly confused between related language varieties like ALG-MSA as they share a lot of words, as well as between varieties that share lexically ambiguous words like FRC-ALG, BOR-ALG, FRC-BOR, NER-ALG, BER-ALG. Several words were neither seen in the training data nor were they covered by the available lexicons which, given that the unknown words are tagged as ALG, leads to confusions such as ENG-ALG, NER-ALG, BER-ALG, BOR-ALG, and FRC-ALG.

(2) a. بويسك قالو غادي يقطعو لما سمانة لجاية راني شريت فاردو **ما** فيه ٢٠ **قرعة سعيدة** تاع ليتر ونص

b. Since they said that they will cut water next week, I have bought a load of 20 bottles of Saida of 1.5 litre.

The MSA-NER confusion is mainly caused by the fact that many NERs are simply common nouns in MSA. For instance, سعيدة could be an adjective in the feminine form in MSA meaning *happy*, or a feminine proper name, or something else. In the context of example (2) it is NER as it refers to the name of a product. The word ما means *water* in ALG, but it is also used as a negation particle in MSA and frequently in ALG, a relative



(a) Average Precision of each model per class (%).

(b) Average Recall of each model per class (%).

(c) Average Loss of each model per class.

(d) Average F-score of each model per class.

Figure 4.2 Average performance of each model per class.

pronoun in MSA, and a noun meaning *mother* in ALG. Likewise قرعة means *bottle* in ALG, but it also means *contest* or *competition* in MSA.

The F-score and Precision of the Char-level RNN model is lower from the baseline on all classes, and the Recall is better on BOR 64.11% compared to only 49.44% on the baseline, and FRC 72.12% compared to 70.70% respectively. ENG, BER, NER, BOR and FRC are the hardest classes to identify with the following respective Loss values: -9.56,

-6.72, -3.89, -3.57, -2.80, and all classes are confused with ALG, the majority class.

The F-score of the Char-level CNN model is better on SND, MSA, FRC, DIG, BOR, ALG compared to the baseline, but it performs worse on NER, ENG, BER. This could be contributed by the worse Recall on these classes which follows the same trend as the F-score. However, in terms of Precision, the Char-level CNN model performs better on ALG, BER, ENG and SND and worse on the remaining classes, with the same kind of confusions as the baseline.

The F-score of the FastText model is low on all classes compared to the baseline. The same holds for Recall and Precision except on BER where the Precision is better 96.18% compared to 94.51% on the baseline. The model produces the same kind of errors as the previous models, but which are most similar to the Char-level CNN model.

Compared to the baseline, the Lexicon model performs better in terms of the F-score on BOR (80.94 compared to 64.70), ENG (73.72 compared to 49.45), and FRC (83.60 compared to 80.30). However it performs significantly worse on BER (18.31 compared to 80.41). This is likely because of the limited coverage of the lexicons. The results also indicate the bias of the lexicons to those classes that are more difficult to distinguish automatically. On the other hand, in terms of the Recall, the Lexicon model outperforms the baseline on all classes, except on ALG. In terms of the Precision, it is only better on ALG and ENG. The model makes similar errors as the FastText model, only more frequently.

Combining FastText and Lexicon models has a positive effect as the F-score, Recall and Precision increase on all classes, mainly on BOR (F-score of 47.10 to 84.74), ENG (F-score of 32.05 to 70.59) and NER (F-score of 58.90 to 80.35). The combined model makes the same errors as previous models but less frequently.

Overall, the results in this section show that a simple Char-level CNN model outperforms the more complicated baseline system which uses a back-off strategy and extra resources. However the Char-level CNN model performs worse on the minority classes, particularly on NER, ENG and BER.

On the other hand, the other models perform better on the minority classes in terms of Recall, but they perform worse on the remaining classes because of the limited coverage of the lexicons or because of lexical ambiguity. This means that the performance of these models is in complementary distribution. We will explore this observation in the following section.

6.2 Models with Background Knowledge

One possible improvement of the models in Section 6.1 is to inject information from the lexicons and the knowledge encoded in the fasttext to the DNN models. In Table 4.4 we

report the average error rate of only the best performing experiments combining different models and resources.

	Model	Error Rate (%)
1	Char-level RNN + lexicon	8.27
2	Word-level RNN + fasttext	8.20
3	Word-level RNN + fasttext + lexicon	**5,34**
4	Char-level CNN + lexicon	**5.18**
5	Word-level CNN + fasttext	9.75
6	Char-level CNN + lexicon + fasttext	6.23
7	Char-level CNN + lexicon + Bootstrapping	5.23
8	Baseline	9.52

Table 4.4 Average error rate of the models with background knowledge.

The results show that RNN models (with original error rate of 13.38% for Char-level RNN) benefit from both adding the lexicon (1) and the fasttext (2). The gain is even higher when combining both with the Word-level RNN (3). The CNN models behave differently when adding lexicon and fasttext. The Char-level CNN (4) performs best with the lexicon with 3% error reduction. The Word-level CNN (5) performs worse with fasttext compared to basic Char-level CNN introducing a 1.57% increase in the error rate (Table 4.3). Also the Char-level CNN (6) does not benefit from combining lexicon and fasttext. It appears that the latter introduces noise that CNN is sensitive to. Likewise, additional bootstrapped training data does not help the otherwise best performing Char-level CNN + lexicon model (7). This may be also explained by the additional noise in the bootstrapped data.

Figure 4.2 indicates that adding lexicon information has a positive effect on the overall performance of the RNN models. The gain from the lexicons is noticeable on all classes where Precision, Recall and F-score increase, most importantly on BER, BOR, ENG, FRC and NER. The same kind of errors are present as with the previous models but fewer in number. For instance the number of errors between ALG-MSA drops from 1,077 to 724, and between FRC-ALG from 104 to 64.

Adding lexicon information to the Char-level CNN model boosts its overall performance over models not using lexicons. All the classes benefit from the lexicon information and their F-score, Recall and Precision increase, most importantly on the minority classes such as ENG, with the same errors but less frequent.

However, adding fasttext does not improve the performance of the Word-level CNN model. Its average F-score decreases on all classes except on ENG where it increases from 22.76 to 29.91.

Compared with the Char-level CNN + lexicon model, adding fasttext to Char-level

CNN does not have the same positive effect. The only significant gain is an increase in Precision on ENG from 82.59% to 84.79%. Char-level CNN + fasttext + lexicon model performs better than the FastText + Lexicon model. It seems that fasttext does not help the CNN model.

On the other hand, adding fasttext to an RNN boosts its performance. The error rate drops to 13.38% (Char-level RNN) and 8.20% (Word-level RNN). While the Precision of each class improves, the Recall drops on both BOR and ENG classes, by 3.35% and 1.97% respectively. The F-score increases on all classes except on ENG where it drops by 1.76%.

Examining the effect of lexicon and fasttext on the RNN models, we find that the Precision on the minority classes, chiefly BOR, ENG, FRC, NER is higher when adding lexicon (87.10%, 78.77%, 88.36%, and 86.77%) compared to when adding fasttext (73.26%, 66.37%, 84.45% and 78.07%), but the Precision on BER is better when adding the fasttext (96.18% compared to 91.51%). The same trend is observed for Recall where BER is the only class that benefits from fasttext compared to lexicon (70.65% compared to 66.47%). ENG is the class which is most negatively effected when adding fasttext with a drastic decrease of 36.45% (23.62% with fasttext and 60.07% with lexicon), followed by BOR with 18.98% decrease, and NER with 7.54% decrease. The F-scores have the same pattern as the Recall.

A gain of adding lexicon to the Word-level RNN + fasttext model is observed on all classes. While Precision increases on all classes, for example on ENG from 78.77% without the lexicon to 88.04% with the lexicon, it slightly decreases for NER from 86.77% to 85.97% and SND from 99.00% to 98.86%. The Recall and F-score increase on all classes.

The gain from using the bootstrapped data is mainly reflected in an increase in Precision on the minority classes such as ENG, BOR, FRC and NER (93.04%, 96.71%, 96.68% and 93.85% compared to 82.60%, 90.56%, 91.31% and 89.43% respectively without using the bootstrapped data). In terms of Recall, the bootstrapped data only boosts ALG and SND classes. The F-scores of the model trained without the bootstrapped data are better on all classes. The insignificant effect of the bootstrapped data could be attributed to the additional noise introduced by the baseline system.

7 Conclusions and Future Work

We have presented DNN models for detecting code-switching and borrowing for a low-resourced language. We investigated how to improve these models by injecting background knowledge in the form of lexicons and/or pre-trained sub-word embeddings trained on an unlabelled corpus, thus taking advantage of the scarce NLP resources currently

available. The results show that the models behave differently for each class of added knowledge. While adding information from the lexicons markedly improves the performance of all models, adding knowledge in the form of pre-trained sub-word embeddings improves the RNN model more than the CNN model. Bootstrapping does not bring a significant overall contribution to performance of our models which is surprising given the previous reports in the literature. However, it does boost Precision of the minority classes.

One future direction worth exploring is how to deal with the problem of misspellings and spelling variations to reduce the irregularities in non-standardised user-generated data as this appears to have a strong effect on the performance of RNN and CNN models.

Chapter 5

Neural Models for Detecting Binary Semantic Textual Similarity for Algerian and MSA

Wafia Adouane, Jean-Philippe Bernardy, and Simon Dobnik

Abstract

We explore the extent to which neural networks can learn to identify semantically equivalent sentences from a small variable dataset using an end-to-end training. We collect a new noisy non-standardised user-generated Algerian (ALG) dataset and also translate it to Modern Standard Arabic (MSA) which serves as its regularised counterpart. We compare the performance of various models on both datasets and report the best performing configurations. The results show that relatively simple models composed of 2 LSTM layers outperform by far other more sophisticated attention-based architectures, for both ALG and MSA datasets.

1 Introduction

Detecting Semantic Textual Similarity (STS) aims to predict a relationship between a pair of sentences based on a semantic similarity score. It is a well-established problem (Agirre et al., 2012) which deals with text comprehension and which has been framed and tackled differently (Beltagy et al., 2013, 2014). In this work we focus on deep learning approach. For example, Baudis and Šedivý (2016) frame the problem as a sentence-pair scoring using binary or graded scores indicating the degree to which a pair of sentences are related.

Solutions to detecting semantic similarity benefit from the recent success of neural models applied to NLP and have achieved new state-of-the-art performance (Parikh et al., 2016; Chen et al., 2017). However, so far it has been explored only on fairly

large well-edited labelled data in English. This paper explores a largely unexplored question which concerns the application of neural models to detect binary STS from small labelled datasets. We take the case of the language used in Algeria (ALG) which is a low-resourced language with several linguistic challenges. ALG is a collection of local colloquial varieties with a heavy use of code-switching between different languages and language varieties including Modern Standard Arabic (MSA), non-standardised local colloquial Arabic, and other languages like French and Berber, all written in Arabic script normally without the vowels.

ALG and MSA are two Arabic varieties which differ lexically, morphologically, syntactically, etc., and therefore represent different challenges for NLP. For instance, ALG and MSA share some morphological features, but at the same time the same morphological forms have different meanings. For instance, a verb in the 1st person singular in ALG is the same 1st person plural in MSA. The absence of morpho-syntactic analysers for ALG makes it challenging to analyse such texts, especially when ALG is mixed with MSA. Furthermore, this language is not documented, i.e., it does not have lexicons, standardised orthography, and written morpho-syntactic rules describing how words are formed and combined to form larger units. The nonexistence of lexicons to disambiguate the senses of a word based on its language or language variety makes resolving lexical ambiguity challenging for NLP because relying on exact word form matching is misleading.

(1) a. فوت سمانة في **دار** مواليا كي وليت لقيت وليدي **دار** حالة منداك نهار راجلي **دار** فرايو ولا جامي اخليني نبات

b. I spent one week at my parents' **house** and when I came back I found that my son **made** a big mess. After that my husband **changed** his opinion and never allowed me to stay over night (at my parents' house).

(2) a. حنا فلمولود نوجدو طعام **لفطور** ونتوما واش غادي تديرو **غدا**

b. In Mawlid we prepare Couscous for **lunch**, and you what will you prepare (for lunch)?

In many cases, while the same word form has several meanings depending on its context,

different word forms have the same meaning. As an illustration, consider examples (1) and (2) which are user-generated texts taken from our corpus (Section 3.1).

In (1), the same word form "دار" occurs three times with different meanings: "house", "made", and "changed" respectively. Whereas in (2), the different word forms "لفطور" and "غدا" mean both "lunch".

We mention these examples to provide a basic background for a better understanding of the challenges faced while processing this kind of real-world data using the current NLP approaches and systems that are designed and trained mainly on well-edited standardised monolingual corpora. We could, for instance, distinguish the meanings of "دار" in (1) if we knew that the 1st occurrence is a noun and the two others are verbs. Likewise, if we had a tool to distinguish between ALG and MSA, it were easier to detect the meaning of "غدا" as "lunch" in ALG rather than the MSA meaning "tomorrow".

Traditional models for detecting STS cannot be applied on such data because they require existing resources and tools, such as tokeniser, stemmer, PoS tagger, etc. to pre-process the data and extract useful features assuming that the data is correctly spelled (standardised orthography).

Thus using deep neural networks (DNNs) is promising because representations can be learned in an unsupervised way. In particular, when trained end-to-end, inputs are mapped directly to the desired outputs without the need to handcraft features. Nevertheless, this learning approach based on pattern matching requires lot of data to learn useful patterns. Besides there are only a few cleaned and labelled textual corpora available for some languages and creating new ones is labour intensive.

Our contributions are as follows. (i) We introduce a newly built (small) ALG dataset for STS. (ii) We compare the performance of different DNN configurations on this dataset, namely: various combinations of Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), pre-training of embeddings, including a replication of two new state-of-the art attention models. (iii) We test whether increasing the dataset size helps. (iv) We test whether language regularisation helps. For this purpose, we run the same experiments on a regularised and comparable MSA translation of the ALG dataset.

The paper is structured as follows. In Section 2 we briefly review some STS appli-

cations. In Section 3 we describe our experimental setup including data and models. In Section 5 we discuss the results and conclude with our future plans in Section 6.

2 Related Work

Diverse techniques and formalisms have been used to deal with various semantic-related tasks. Among others, machine learning has been applied to detect semantic textual relatedness such as Textual Entailment (TE) (Nielsen et al., 2009), STS (Agirre et al., 2016), Paraphrase Identification (PI) (Liang et al., 2016), etc. Earlier systems use a combination of various handcrafted features and are trained on relatively small datasets. For example, Dey et al. (2016) uses Support Vector Machines with a set of lexical, syntactic, semantic and pragmatic features. As discussed earlier, these features are not available from our dataset.

These tasks have recently attracted more attention when DNNs became practical, mainly due to the availability of large labelled datasets such as the Stanford Natural Language Inference corpus (SNLI) containing 570K sentence pairs (Bowman et al., 2015), Sentences Involving Compositional Knowledge (SICK) containing about 10K sentence pairs (Marelli et al., 2014), the Microsoft Research WikiQA Corpus (WIKIQA) containing more than 23K sentence pairs (Yang et al., 2015), the Quora dataset released by Kaggle competition consisting of 400K potential question duplicate pairs[1], and the Microsoft Research Paraphrase (MSRP) consisting of more than 5K sentence pairs (Dolan and Brockett, 2005).

We follow the approach of Baudis and Šedivý (2016) who consider that several tasks dealing with detecting semantic relatedness are technically similar and can be formulated as sentence-pair scoring. They propose a generic framework for text comprehension for evaluating and comparing existing systems. Several DNN systems have been proposed. For instance, Mueller and Thyagarajan (2016) propose a siamese recurrent architecture using Manhattan LSTM (MaLSTM) for STS. They use word embeddings supplemented with synonymy information, LSTM and Manhattan distance to compose sentence representations.

Additionally, complex DNN systems with various attention mechanisms have been proposed to deal with more than one semantic similarity task at the same time. For instance, Yin et al. (2015) apply attention to represent mutual influence between the input sentence pairs. Similarly, Parikh et al. (2016) propose the Decomposable Attention Model (DecompAtten) which relies on alignment using neural attention to decompose the task of natural language inference into sub-tasks which are aggregated and used to predict

[1]Corpus webpage: https://www.kaggle.com/quora/question-pairs-dataset

the output. In the same direction, Chen et al. (2017) propose the Enhanced Sequential Inference Model (ESIM) composed of a bidirectional LSTM (BiLSTM) encoder, and a soft alignment which computes attention weights to determine the relevance between two input sentences. Then they use another BiLSTM layer to compose local inference information and aggregate the output by applying average and max pooling, and concatenating all in one vector.

All preceding models involve considerable sophistication of design and sometimes require specific dataset labelling. This is to say they are normally trained on large well-edited and labelled datasets that are available for English but are unavailable for most other languages. Unlike the previous work, we will compare the performance of two presumably best performing architectures to simpler architectures similar to MaLSTM but with different additional components on a small unedited dataset.

3 Datasets

3.1 ALG STS data

To the best of our knowledge, there is no ready-to-use ALG data for any semantic similarity related task prior to this work. As a basis we use an extended version of the ALG unlabelled dataset (Adouane et al., 2018a) which currently contains 408,832 unedited short colloquial texts (more than 6 million words) collected from online discussion forums. For the STS task we created a dataset of 3,000 sentence pairs as follows. We randomly selected 1,000 sentences from the ALG unlabelled data, including various topics and text lengths. We asked two ALG native speakers to produce for each given sentence is semantically equivalent and the other can be semantically similar but not equivalent, i.e., it could include the same words or could be about the same topic.

(3) a. لالا ماشي باهية الروز قديم دوكا .
الوردي ما عجبنيش ما هوش الامود .

b. No, it is not beautiful, pink is outdated.
I do not like pink, it is not fashionable.

(4) a. هديت ليما تارت تاع الشوكو .

عجبتني لاتارت تاع الشوكو لي دارتها يما .

b. I offered to my mother a chocolate pie.

I like the chocolate pie that my mother baked.

In example (3), the two sentences are semantically equivalent but in (4) the two sentences are roughly about the same topic and include "chocolate pie", "mother" and "I" but some important information differs — like who did what.

The annotators were free to use whatever words as long as the produced sentences sounded natural to them and the above instructions were respected. We provided them with two examples of the desired sentences and explained the difference. We combined all the sentences and created 3,000 unique sentence pairs.

In the second part of dataset creation, we asked three different native speakers to provide a similarity score between 0–5 for each sentence pair following the guidelines used in the SemEval-2016 shared task (Agirre et al., 2016). Finally, another annotator performed manual checking and majority voting of the labels.

Because the annotators assigned scores according to their judgement, the resulting data is not balanced in terms of the number of instances per class (0–5) as shown in Table 5.1. The corpus contains 36,767 words, 7,074 unique words and sentence average length of 5.19 words or 34 characters.

Score	Interpretation	#Pairs
0	The two sentences are completely dissimilar.	1,550
1	The two sentences are not equivalent, but are on the same topic.	237
2	The two sentences are not equivalent, but share some details.	140
3	The two sentences are roughly equivalent, but some important information differs.	63
4	The two sentences are mostly equivalent, but some unimportant details differ.	16
5	The two sentences are completely equivalent, as they mean the same thing.	994

Table 5.1 Labelling guidelines and statistics about ALG STS dataset.

We first tried to predict the graded six similarity scores as multi-class STS, but the

systems (Section 4) only predicted the most frequent classes, namely scores 0 and 5. This behaviour suggests that given the size of the dataset and the number of instances for each class, the classes are not distinguishable enough. Therefore, we re-framed the task as a binary STS: either two sentences are semantically equivalent or not, rather than predicting their graded similarity (Agirre et al., 2015; Xu et al., 2015). To this end, we merged all scores which do not capture semantic equivalence (0 to 4) into a single class, and refer to them as non-equivalent. The remaining score of 5 stands on its own as completely equivalent. The resulting binary labelled data contains 994 equivalent sentence pairs and 2,006 non-equivalent sentence pairs.

3.2 MSA STS data

Contrary to ALG, MSA is a well-represented Arabic variety with standardised spelling. We use a large MSA Wikipedia corpus[2] consisting of more than 52 million tokens. We automatically removed all words written in non-Arabic script and punctuation. We refer to this corpus as MSA unlabelled data.

We also created a labelled STS corpus for MSA by commissioning another pair of ALG native speakers to faithfully translate the ALG STS dataset into MSA. They were instructed to keep the order of words and structures as close as possible to the ALG sentences without changing the meaning. We manually checked the quality of the translation, corrected some minor misspellings and checked the corresponding similarity scores (0–5). We proceeded in the same way as for ALG and created a binary MSA STS dataset including equivalent and non-equivalent sentence pairs.

Both binary and multi-class STS MSA datasets have the same number of sentence pairs as their ALG corresponding datasets. However, the MSA datasets have a smaller vocabulary, consisting of only 5,527 unique words from a total of 37,832 words. The average sentence length is 6.84 words or 33.26 characters. The difference in the vocabulary size is mainly due to misspellings and spelling variations in the ALG corpus: it is non-standardised language. Yet both ALG and MSA datasets have relatively short sentences and they are about the same topics since one is a translation of the other.

4 Models

All models have the same basic structure. They consist of two identical siamese networks, one for each input sentence as shown in Figure 5.1. The main differences between the models are in the embeddings, the sentence encoder, the distance measure, and the objective function for the final prediction. Note that using the same colour for "Sentence

[2]The MSA corpus was downloaded from: http://goo.gl/d7pxZb.

Encoder" is meant to show that the trained parameters are shared between the left (1) and right (2) part of the network.

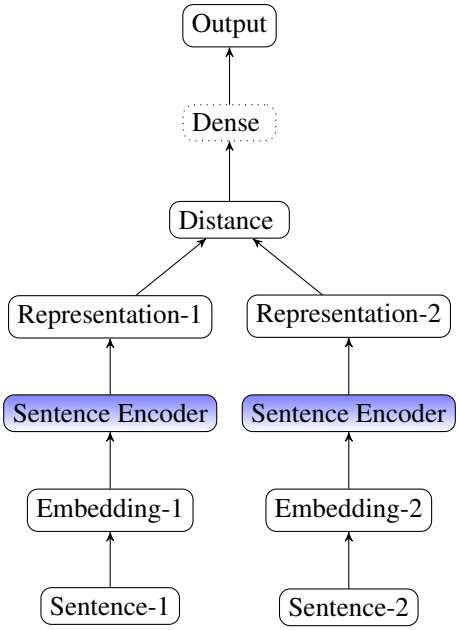


Figure 5.1 Siamese network architecture.

4.1 Embeddings

We use two kinds of embedding layers. First, an embedding layer trained only on the training data based either on characters or words, initialised either with a uniform or a normal distribution. We refer to these embeddings as trainable as a contrast to pre-trained embeddings. Second, we pre-trained a word2vec and FastText embeddings on the larger unlabelled data mentioned in Section 3, using the Gensim (Řehůřek and Sojka, 2010) and FastText (Bojanowski et al., 2016) libraries. For word2vec embeddings, we used a context size of 5 words, minimum occurrence of 1 and dimension of 300. For FastText embeddings, we used dimension of 300, range of sub-characters between 3-5 characters, and a context size of 5 words, and training for 200 epochs. The goal of using pre-trained word embeddings is to test whether we can make use of the large unlabelled corpora.

4.2 Sentence Encoders

We use either an RNN or a CNN with different configurations to encode each sentence and output a representation for each. The sentence encoders are identical for both sentences and share weights. Here are some of the encoders that we experimented with.

- **RNN-based encoder** consisting of a stack of standard and/or bidirectional LSTM layers with 300 units and a dropout rate of 3%.

- **CNN-based encoder** consisting of a stack of convolution layers with 60 filters of size 5, with a ReLU activation and a dropout rate of 10%, followed by max pooling with a pool size of 3, followed optionally by a global average pooling and global max pooling multiplied together.

- **CNN-RNN-based encoder** A combination of RNN and CNN encoders where we stack a number of convolution layers with 60 filters of size 5, with a ReLU activation and a dropout rate of 10%, followed by max pooling with a pool size of 3 and a number of RNN layers (either standard or bidirectional LSTMs).

- **Attention-based encoder** Roughly put, the idea of an attention mechanism is to attend to some parts of an input/output when deriving its representation (Bahdanau et al., 2014). We implement the Decomposable Attention (DecompAtten) and Enhanced Sequential Inference Model (ESIM) models, as described in Section 2.

4.3 Distance

The distance component serves to compose the sentence representations. We use standard distances such as Euclidean distance, Manhattan distance, and Cosine similarity.

4.4 Dense

Instead of using a distance measure between the sentence representations, we compose the two sentence representations by multiplication (multp), subtraction (subtr), summation (sum), or concatenation (conct) as in the ESIM model. This operation is followed by a dense layer. We indicate that this layer is optional by using a dotted frame in Figure 5.1. When it is used, we use a sigmoid activation with a binary cross-entropy loss.

Except for the pre-trained embeddings, all models are trained end-to-end for 300 epochs using a batch size of 64 and Adam optimiser with a learning rate of 0.001.

5 Experiments and Results

We randomly selected from the binary ALG STS dataset 250 sentence pairs of each class (equivalent and non-equivalent) as the test set (500 in total), 200 sentence pairs as a development set, and the remaining 2,300 sentence pairs as a training set. Note that balancing the test set is not essential. Likewise, we split the binary MSA STS data by taking the corresponding translations for each instance in the ALG dataset.

The hyper-parameters reported in Section 4 were selected based on the reported common values in the literature for similar tasks and fine-tuned on the development set. Moreover, because of the stochastic nature of the neural models where the results vary between each training run, we report the average performance on the test set over 10 training runs for the best performing models trained on both training and development data following Baudis and Šedivý (2016) and Yin et al. (2015).

In order to increase the size of the training data and to boost the instances of the minority class (equivalent sentence pairs), we duplicated equivalent sentence pairs by reversing their order so that each sentence pair appears only once in the same order. This is a standard data augmentation practice used to mitigate the limited availability of labelled training data (Yin et al., 2015; Mueller and Thyagarajan, 2016). The augmented training set contains 3,244 sentence pairs (1,488 equivalent and 1,756 non-equivalent pairs). Because there is no previous work reported for ALG on a similar task, we resort to the binary random guess, namely 50% as a baseline. We report the overall accuracy for the same models with and without the augmented training data, for both ALG and MSA separately. In Table 5.2, we only report the models that outperform the baseline. Note that Acc is accuracy with non-augmented training data and Acc-aug is accuracy with the augmented training data.

					ALG		MSA	
	Model	Emb	Encoder	Dist	Acc	Acc-aug	Acc	Acc-aug
1	char-RNN	trainable	2-LSTM	multp	55.78	61.84	59.65	67.80
2	char-RNN	trainable	2-LSTM	subtr	70.38	78.56	69.02	71.37
3	word-RNN	trainable	2-LSTM	multp	**85.06**	87.20	**85.19**	86.69
4	word-RNN	trainable	2-LSTM	subtr	73.73	**92.76**	68.90	88.20
5	word-RNN	word2vec	2-LSTM	subtr	71.40	92.51	67.86	**89.46**
6	word-RNN	FastText	2-LSTM	subtr	71.68	92.70	68.06	88.57
7	word-CNN	trainable	1-CNN	sum	50.00	50.00	50.00	50.00
8	DecompAtten	trainable	attention	sum	50.44	53.00	50.02	50.44
9	ESIM	trainable	attention	conct	52.34	52.80	50.34	50.39

Table 5.2 Average accuracy (%) of the models.

5.1 Binary STS for ALG

5.1.1 Non-augmented data

The results show that char-RNNs composed of 2 standard LSTM layers and trainable embedding layer with normal distribution (1) and (2) perform worse than their word-based counterparts (3) and (4). This result contradicts the conclusion that character models are better at modelling morphologically rich languages (Vylomova et al., 2017), and consequently they are better in dealing with misspellings and capturing spelling variations.

The best performance is achieved by a word-based 2-LSTM layer encoder and a trainable embedding layer (3), using multiplication as a distance with an accuracy of 85.06%. Nevertheless, char-RNN performs better with subtraction rather than multiplication as a distance (2). Adding pre-trained embeddings word2vec (5) and FastText (6) to the word-level RNN in (4) decreases the accuracy by 2.33 and 2.05 points respectively. This effect could be caused by the noise in the ALG unlabelled data on which the embeddings were trained.

A 1-layer CNN with no pre-trained embeddings and using summation of the sentence representations as a distance (7) performs the best compared to the other options with CNN encoder but overall it performs quite poorly. Likewise combining 1-CNN and 1-LSTM layers as encoder (not shown in Table 5.2) does not have an effect over using only 1-CNN layer. The models predict all the test sentence pairs as non-equivalent. In other words, the network could not learn enough to properly distinguish between the two classes.

These results contrast those reported by Kadlec et al. (2015), namely that CNN models perform better with little data compared to RNN models. However, it is hard to quantify what is considered to be small apart from the number of examples. In general, neural models learn useful features when they are trained on enough representative data. That is to say it is not just a question of data size, but it is more about the complexity of the features and the functions that they should learn. In our case, we suspect that the sparsity and the noise in the data is making learning harder for CNN models.

Regarding attention-based encoders, ESIM (9) outperform DecompAtten (8), and both perform slightly better than the baseline.

The poor performance of these models with little noisy data could be related to the fact that attending to some parts of a sentence or focusing on surface form similarity is misleading since the same word form can have different meanings and different word forms can have the same meaning, especially that the data does not contain named entities or punctuation or digits which could help alignment.

		Equivalent			Non-equivalent		
	Model	Precision (%)	Recall (%)	F-score	Precision (%)	Recall (%)	F-score
1	char-RNN-multp	73.91	53.54	62.10	63.12	80.80	70.88
2	char-RNN-subtr	88.02	66.54	75.78	72.76	90.80	80.78
3	word-RNN-multp	86.96	88.00	87.48	87.85	86.80	87.32
4	word-RNN-subtr	89.67	97.20	93.28	96.94	88.80	92.69
5	word-RNN-word2vec	89.30	96.80	92.90	96.51	88.40	92.28
6	word-RNN-FastText	**90.84**	95.20	92.97	94.96	**90.40**	92.62

Table 5.3 Average performance of the models on the ALG augmented data.

5.1.2 Augmented data

All models benefit from the augmented data, except word-CNN (7) for which the gain is not clear. The performance of the char-RNN (2) shows 8.18 point improvement in accuracy. This result supports the hypothesis that the poor performance of the model trained on the non-augmented data is caused by the small size of the sparse noisy data which makes it hard for the char-RNN to learn useful patterns.

Yet the significant improvement of the word-RNN (4) by 19.03 points, indicates that word-RNN suits better our case.

Models with subtraction as a distance benefit the most from the added data. Similar to their behaviour on non-augmented data, adding pre-trained embeddings slightly decreases the performance of the model compared to not adding them.

Comparing embeddings, word2vec causes slightly more drop in the performance of word-RNN compared to FastText.

Attention-based models benefit also from the added data, but the gain is larger for DecompAtten compared to ESIM.

Looking at the performance of the models for each class shown in Table 5.3, it is clear that the RNN models are doing quite well for both classes whereas CNN and Attention-based models, not included for space limits, are too biased to the non-equivalent class. Figures in bold are meant to highlight the gain due to pre-trained embeddings.

Error analysis of the word-RNN model (4) shows that 7 equivalent sentence pairs are misclassified as non-equivalent and 28 non-equivalent sentence pairs are misclassified as equivalent. We manually checked the errors and found that most of the non-equivalent pairs misclassified as equivalent have at least one word in common as in example (5) but the words have a different meaning depending on their context. However, distinguishing between word senses is hard because the context is not entirely sufficient. Example (6) is an equivalent pair misclassified as non-equivalent. The common pattern among the misclassified examples is that they have no exact words in overlap. This could explain why attention-based encoders, with some form of alignment, fail to generalise to new

instances. Probably there is a bias to the form with one meaning when senses are not sufficiently differentiated.

(5) a. . شفت حجا بيزار

. سي بيزار ما شفتوش

b. I saw a weird thing.

It is weird that I did not see it.

(6) a. . راني نحم وقتاش تدخل لبورس

. يادرى هذيك المنحة وينتا تجي

b. I am thinking when the grant will be received.

I wonder when the grant will be paid.

5.2 Binary STS for MSA

We now evaluate the performance of the same DNN configurations on parallel regularised MSA data using the same hyper-parameters as in Section 5.1. The results are reported in Table 5.2.

5.2.1 Non-augmented data

Again, the word-RNN with multiplication (3) performs the best with an accuracy of 85.19%. The char-RNN (1) with the same settings achieves an accuracy of only 59.65%. Using subtraction, the char-RNN (2) slightly outperforms the word-RNN (4), with 69.02% and 68.90% accuracy respectively.

		Equivalent			Non-equivalent		
	Model	Precision (%)	Recall (%)	F-score	Precision (%)	Recall (%)	F-score
1	char-RNN-multp	69.86	61.20	65.25	65.48	73.60	69.30
2	char-RNN-subtr	76.35	62.25	68.58	67.92	80.57	73.70
3	word-RNN-multp	87.04	86.00	86.52	86.17	87.20	86.68
4	word-RNN-subtr	85.77	91.60	88.59	90.99	84.80	87.78
5	word-RNN-word2vec	**87.17**	**92.77**	**89.88**	**92.21**	**86.23**	**89.12**
6	word-RNN-FastText	**86.97**	91.16	**89.02**	90.64	**86.23**	**88.38**

Table 5.4 Average performance of the models on the MSA augmented data.

Adding FastText (6) and word2vec (5) pre-trained embeddings causes the accuracy of the best word-RNN (4) of 68.90% to decrease slightly to 68.06% and 67.86% respectively.

This could be due to the embeddings not distinguishing between the different senses of the same word, i.e., output one vector representation for each word form. Also the large MSA corpus on which the embeddings were trained can have different topical distribution than the MSA STS data. As with the ALG data, CNN (7) and attention-based encoders (8–9) behave the same.

5.2.2 Augmented data

Trained on augmented data, models with subtraction yield the best performance compared to multiplication, and word-RNN (4) outperforms char-RNN (2) with 88.20% and 71.37% accuracy respectively. Unlike when using the ALG data, pre-trained embeddings improve slightly the performance of (4) with 0.37 (6) and 1.26 (5) points gain in the error reduction respectively. The positive effect of the pre-trained models could be due to the fact that more regularities are captured. Training on augmented MSA data does not yield any significant gain over training on non-augmented data for CNN (7) and attention based models (8–9).

In Table 5.4 we report the performance of each model per class. Due to space limits, we do not include the CNN and attention-based models which are again struggling with the equivalent class and are biased towards the non-equivalent class. The gain from the pre-trained embedding is in bold. The models perform almost the same for both classes but slightly worse than with the ALG data.

Example (7) is a non-equivalent sentence pair misclassified as equivalent, and example (8) is an equivalent pair misclassified as non-equivalent by the word-RNN model (5).

(7) a. الكيكة أنا أيضا جربتها والله روعة وجدت أولادي كلهم لعقوها .

جربتها كم من مرة كانت سامة وحاسدة .

b. I also tried the cake and it was great, I discovered that my kids finished it.

I tested her many times and she was jealous and envious.

(8) a. يا ليتهم يغيرون هذه **المذيعة** .

يا ريت يغيرون لنا هذه **المنشطة** .

b. Wish they change this presenter.

Hope they will replace this presenter.

It is hard to explain why these examples are misclassified, except that there is not enough context to discover the meaning of the words. For instance, in (8) the words in bold “مذيعة” , “منشطة” are synonyms in these two sentences, and the two sentences have two

more word overlaps "هذه" and "يغيرون" with the same meaning. This should help classifying the two sentences as equivalent, but it is not the case possibly because their contexts are different.

6 Conclusions and Future Work

We have presented a new STS dataset for ALG user-generated short texts and its MSA translation. We then described the neural network models trained end-to-end with different configurations and compared their performances on a binary STS task. The results show that relatively simple model architectures, composed of two word-based LSTM layers with subtraction as explicit similarity measure used in the training task, suit better our data compared to the other more sophisticated architectures which might require more data to achieve better performance.

We ran the same experiment on the MSA data, but the results were not really different from the ALG data. However, pre-training embeddings performed better with MSA, probably because the language is more regular and knowing some structure ahead helps. The performance improved with more data for the minority class (equivalent sentence pairs) for both ALG and MSA. However, surprisingly the gain of some models with ALG is greater than their gain with MSA. This is probably caused by the noisiness and the sparsity of the data, the linguistic differences between MSA and ALG, the data size, or all these factors together. Further and deeper experiments and analyses are needed for a better understanding of the results.

Overall, the results of the end-to-end training are promising and could be generalised to other related languages or language varieties with the same low-resource settings.

As a future work, we want to explore ways to improve the learning capability of neural models from small noisy datasets without handcrafted features, for example by reducing the noise in the colloquial data (ALG) by normalising spelling variation.

Chapter 6

Normalising Non-standardised Orthography in Algerian Code-switched User-generated Data

Wafia Adouane, Jean-Philippe Bernardy, and Simon Dobnik

Abstract

We work with Algerian, a low-resourced non-standardised Arabic variety, for which we compile a new parallel corpus consisting of user-generated textual data matched with normalised and corrected human labels following data-driven and our linguistically motivated standard. We use an end-to-end deep neural model designed to deal with context-dependent spelling correction and normalisation. Results indicate that a model with two CNN sub-network encoders and an LSTM decoder performs the best, and that word context matters. Additionally, pre-processing data token-by-token with an edit-distance based aligner significantly improves the performance. We get promising results for the spelling correction and normalisation, as a pre-processing step for downstream tasks, on detecting binary Semantic Textual Similarity.

1 Introduction

Natural language processing (NLP) research has achieved impressive results, notably thanks to the use of deep neural networks (DNNs) which has pushed the field forward, achieving unprecedented performance for various tasks. However, research is often focused on large, standardised, monolingual and well-edited corpora that exist for a few well-resourced languages. We believe that such corpora will not generalise to all languages and domains, particularly regarding the colloquial varieties used in new communication channels. In fact, the large unstructured data coming from such channels is not

only unedited, it also poses serious challenges to the current NLP processing pipelines and approaches as a whole.

Traditionally, the *standard language ideology* has dominated linguistic studies: it has been frequently assumed that languages are naturally uniform and monolingual. Nevertheless, the new online data reveals that standardisation is neither natural nor universal, it is rather a *human invention* (Milroy, 2001), and variation is the norm. This variation presents several challenges to studying and processing dialects in social media (Jørgensen et al., 2015). These challenges are even more pronounced in multilingual societies where people use more than one language or language variety at the same time. We consider the case of the colloquial language used in Algeria (hereafter referred to as ALG) which combines both linguistic challenges mentioned above: (i) it is non-standardised, and (ii) it is a mixture of languages which involves code-switching between Modern Standard Arabic (MSA) and local Arabic, French, Berber, and English. (We refer the interested reader to the work of (Adouane et al., 2018a), who provides an overview of the linguistic landscape in Algeria.)

In interactive scenarios, people usually use spoken-like language and spontaneous orthography which reflects local variations. Our observations confirm those of Eisenstein (2013), namely that speakers have some-kind of tacit knowledge of spelling which is not completely arbitrary. However, it is hard to distinguish between local varieties and draw a clear borderline between them due to the free mobility of people, their ability to interact online, and the fact that these varieties are closely related and therefore hard to describe formally. Therefore, we find that using location to map dialectal variation (Doyle, 2014) is not useful. In many cases, the spelling is not consistent even by a single person within the same conversation. There is nothing intrinsically wrong with this inconsistency for there is no standard form to take as a reference. Besides, spelling variation does not hinder mutual understanding.

Current NLP approaches based on learning underlying regularities from data is not suitable to sparse noisy data. Furthermore, the data written in Arabic script is already rich in orthographic ambiguity because vowels are not written, except in very specific settings. Our focus is to process such user-generated textual data, reflecting the real use of a language. Therefore, for computational purposes, we want to automatically reduce the data sparsity caused by spelling inconsistency by normalising it based on spelling decisions that we designed, and build a tool that can be used for pre-processing such texts for other NLP tasks.

This paper is an attempt to take advantage of DNNs to reduce spelling inconsistency by performing several transformations (normalisation, disambiguation, etc.) detailed in Section 3 as a single machine-learning task. It is significantly different from the well-established spelling error correction mainly because we have to deal with a non-

standardised code-switched language. In addition to the fact that ALG is a low-resourced language with respect to the size, quality and the diversity of the available labelled data, it suffers from the absence of other tools and linguistic resources required by current NLP techniques such as tokenisers, syntactic parsers, morphological taggers, lexicons, etc.

As contributions, (i) we introduce a new user-generated corpus for ALG with its parallel spelling normalised and corrected version produced by human annotators. (ii) We describe our spelling decisions aiming to reduce orthographic ambiguity and inconsistency for NLP tasks. These decisions are not the only possible ones, and can be debated and further refined. (iii) We propose a general end-to-end model for context sensitive text normalisation of non-standardised languages. We opt for end-to-end deep learning approach (with only a simple automatic pre-processing) because it is not only expensive and time consuming to build equivalent rule-based tools from bottom up, but it is also hard to exhaustively define spelling norms given the high linguistic variation.

The paper is organised as follows. In Section 2 we survey related work. In Section 3 we present our newly compiled parallel corpus and explain our data processing decisions. In Section 4 we give information about data statistics and alignment. In Section 5 we describe our models. In Section 6 we describe our experiments and discuss the results. We conclude in Section 7 with potential future improvements.

2 Related Work

The task of normalising user-generated non-standardised data is closely related to the one of historical text normalisation (Pettersson, 2016), namely they present similar challenges for the current NLP – little sparse data. While the latter has a standardised spelling as a reference, the former does not because many colloquial languages have not undergone the standardisation process. Bollmann (2019) surveys the approaches used for historical text normalisation for a set of languages. Both tasks are mainly framed as (statistical/neural) machine translation mostly at a token level where the source and the target language are the same or a standardised version of one another.

Similarly to the previous work, we formulate our task as a sequence-to-sequence (seq2seq) learning problem, but in contrast we take word context into account. A large body of work has been done to address the problem of seq2seq prediction and has achieved impressive results for diverse NLP tasks. Encoder-decoder models are most frequently used for seq2seq prediction with varying the architectures of the encoder like Recurrent Neural Network (RNN) in (Cho et al., 2014a; Sutskever et al., 2014), bidirectional Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) in (Bahdanau et al., 2014), Convolutional Neural Networks (CNN) in (Vinyals et al., 2015).

Our CNN-based architecture (see Section 5) is reminiscent of what has been proposed

for machine translation by Gehring et al. (2017) but instead they use CNN for both encoder and decoder with multi-step attention. A difference with our model is that we use two sub-networks (LSTM/CNN and CNN/CNN) as an encoder, jointly trained to learn contextual representations of words. Then we use an LSTM as decoder instead of a CNN. Compared to the model of Bahdanau et al. (2014), an important difference is that we do not jointly train alignment and seq2seq prediction. Instead we perform alignment separately as a pre-processing step using edit-distance.

None of the mentioned models have been tested on the same prediction task as ours or on a related language. As the most closely related work for spell checking, Ghosh and Kristensson (2017) propose a seq2seq neural attention network system for automatic text correction and completion. They combine a character-based CNN and a Gated Recurrent Unit (GRU) (Cho et al., 2014a) as encoder and a word-based GRU as decoder using a 12 million word English corpus. Recently, Sooraj et al. (2018) employed a character-based LSTM language model to detect and correct spelling errors for Malayalam. In the same line of research, Etoori et al. (2018) propose an attention model with a bidirectional character-based LSTM encoder-decoder trained end-to-end for Hindi and Telugu spelling correction using synthetic datasets.

Contrary to the task we are trying to address in this paper, the mentioned work deals either with spelling correction for monolingual standardised languages or historical text normalisation for standardised languages. This makes our task linguistically more challenging because our data includes more languages hence the model has to find the correct spelling of a word not only based on its context but also based on its language.

There has been work done for Arabic automatic error correction mainly for MSA including the work of Shaalan et al. (2012) and others included in the Arabic shared task (Mohit et al., 2014). Still they are inadequate to process non-standardised Arabic varieties given the significant phonological, morphological and lexicon differences between MSA and Arabic dialects (Watson, 2007). To the best of our knowledge, this is the first effort to process user-generated non-standardised dialectal Arabic textual data end-to-end.

3 Data Preparation

3.1 Corpus creation

As a basis we take the extended version of the unlabelled dataset of (Adouane et al., 2018a). Our extended version of it consists of 408,832 automatically pre-processed user-generated short texts from social media, such as forum discussions, and contains more than 6 million words. The automatic pre-processing involves removal of punctuation, emoticons and reduction of repeated letters to a maximum of two. Indeed, Arabic orthog-

raphy does not use more than two adjacent occurrences of the same letter, and repeats in social media texts are mainly typos or emphasis. For this work, we further pre-processed this dataset by removing any existing diacritics representing short vowels because they are used rarely and inconsistently, even in the texts generated by the same user. We assume that such idiosyncratic variation will not affect our task in terms of semantics and bring about more robustness to language processing, especially because diacritics are not commonly used outside of the formal register. We also normalised many commonly used (often french-based) Latin script abbreviations to their full versions using the most frequent spelling in Arabic script including psk/because, r7/recipe, bnj/good morning, b1/well, 2m1/see you tomorrow, dsl/sorry, on+/moreover, tj/always, etc.

All texts are written in Arabic script and display spelling variations, typos and misspellings wrt. MSA, diglossic code-switching between MSA and local colloquial Arabic varieties, bilingual code-switching between Arabic varieties; French; Berber and English. From this further pre-processed unlabelled dataset, we created a parallel corpus of manually normalised texts. For this purpose, we randomly selected 185,219 texts and had 5 human annotators, who are native speakers with (computational) linguistics background, to edit and process them. The process took 6 months mainly working on lexical and syntactic ambiguities which require linguistically informative decisions, and all annotators checked the labels of each other. We give here a few examples of spelling variation, but the corpus contains 50,456 words and 26,199 types to be normalised or corrected. Note that we will use word to refer to lexical words and tokens to refer to lexical words plus digits and interjections.

3.2 Labelling standard

In order to guide the annotators in producing parallel normalised text, we designed the following labelling standard which involves (i) spelling correction and (ii) spelling normalisation tasks.

3.2.1 Spelling correction for MSA

Misspelled MSA words are corrected using MSA orthography based on their context. نضيف، غذاءيه، جزاءر، مناقشه (clean, nutritional, Algeria, discussion) are corrected as نظيف، غذائية، جزائر، مناقشة.

3.2.2 Typographical error correction

The texts have been written on different kinds of keyboards resulting in lot of typos which mainly include missing spaces like in ومخلوهاشتخرجو or additional spaces like in لعا يلة which have been respectively corrected as وماخلوهاش تخرج و (and they did not let her to go out and) and لعايلة (the family). There are also keyboard related typos like reversing the order of letters or substituting one letter by another like in ولبدي where ب should be replaced by ي to get the correct intended word وليدي (my son).

These typos can be detected from their context by manual checking. Usually they are not valid words and tend to be consistently generated by the same user which suggests that they may be related to their typing style and conditions. In خيهيا فغيهيا the user used the same wrong letter ى twice instead of ر and the correct form is خيرها فغيرها (the better is in something else).

3.2.3 Spelling normalisation

Non-MSA words including local Arabic varieties, French, Berber, English and neologisms are spelled spontaneously in Arabic script where users use improvised phonetically-based orthography.

- **Local Arabic varieties** To deal with the spelling variation in colloquial varieties, a conventional orthography for dialectal Arabic (CODA) has been proposed for Egyptian (Eskander et al., 2013) and has been extended for Algerian (Saadane and Habash, 2015) and recently for several other Arabic varieties (Habash et al., 2018). We share the overall goals with the authors of CODA that a conventional orthography for developing NLP tools should preserve phonological, morphological and syntactic information of dialectal texts, should not diverge substantially from the existing forms, and should be easy to learn and write by the annotators.

  However, CODA is primarily a recommendation of guidelines with several open questions related to how these guidelines could be implemented in new scenarios. In our case the most relevant open question is how to deal with multilingual code-switched data found in ALG. Using the existing recommendations from CODA would be in several cases impractical because several phonological distinctions required by the varieties in ALG could not be encoded and would have to be listed as exceptions.

In other cases, the application of CODA would also require a substantial rewriting of the original user-generated text. Instead we use data statistics as heuristics to find the canonical forms. We first train word embeddings using FastText (Joulin et al., 2016) on the entire unlabelled data. We collect a list of all words in the corpus and for each word we use FastText to predict the 10 most similar words and their frequencies. This normally returns the spelling variations of that word. A human annotator then decides whether the returned cluster should be considered as a spelling variation and assigns the most frequent spelling as the canonical form for all word occurrences in this cluster.

This is not a trivial task to be performed fully automatically because the model often returns unrelated words for less frequent words (case of the majority of words in the dataset). Hence a human expertise is needed. Contrary to CODA where every word has a single orthographic rendering, if a word has more than one frequently occurring spelling we keep such variations because they reflect local lexical or phonological differences which may be useful for sociolinguistic studies. For example, we keep both spelling variations of question words قداش، قداه and علاش، وعلاه (when and why) because they occurred very frequently and could be mapped to the same form if needed.

In cases where the difference between MSA and local Arabic spelling of a word is based on phonetically close sounds such as the sounds س [s] and ص [s^{ς}] as in سمعة، صمعة (reputation) or between ت[t] and ط [t^{ς}] as in تريق، طريق (road), and the meaning is preserved, MSA spelling is used. These cases are hard to identify automatically and require human expertise. Making spelling MSA-like as practically as possible will facilitate the reuse of existing MSA resources. Nevertheless, in cases where a word does not exist in MSA and has several different spellings, the most frequent one is used provided that it is not homonymous with another existing word. Such words include frequent local Arabic words like امالا ، ضك ، علاجال (so, now, for) with 27, 59 and 39 spellings respectively, along with the newly created words like نريض (I practise sports) and نرمضن (I fast).

- **Non-Arabic words** The dataset includes French, Berber and English words, and the limitation of the Arabic script creates more ambiguity regrading how to spell non-existing sounds like /g, p, v/. The most frequent spelling with long vowels is used. For

example, the French word “journal” (newspaper) occurs with 6 spellings all mapped to جورنال which is the most frequent spelling.

3.2.4 Word sense disambiguation

Using various languages with spelling variation at the same time creates considerable ambiguity, especially when all the varieties are written in the Arabic script. One particular frequent source of lexical ambiguity concerns the spelling of the French definite articles (le, la, les) spelled as لو ، لا ، لي, either separated or concatenated to the word they associate with. However, the Arabic spelling is ambiguous because each of the above words means something else in MSA or local Arabic. For instance, لي when written as a separate word could either be a prepositional phrase (for me) in MSA or a relative pronoun (who / that / which) in local Arabic. For this reason we decided to spell French definite articles attached as prefixes like the Arabic definite article ال. This allows disambiguation of cases like: لي ماش (hair strand dyeing) in French and (who is not) in local Arabic.

The Berber word for “window” is spelt as طاقة which means energy in MSA. Since Berber does not have a standardised spelling in Arabic script[1], we decided to change the spelling to تاقة which is another spelling found in the dataset. Furthermore, lexical ambiguity is caused by the absence of sounds (and corresponding graphemes) in Arabic like /g,v,p/. “Group” is spelled : غروب ، قغوب ، قروب ، جروب ، قرووب where غروب and قروب mean “sunset” and “closeness” in MSA. To disambiguate these senses قغوب is used for “group”.

3.2.5 Negation particle

The various spellings of the word ما cause significant lexical and syntactic ambiguity. When written separately, it could be a relative pronoun or an interjection in MSA, a feminine possessive pronoun in French, ’mother’, ’water’ or a negative particle in local Arabic. We decided to spell this negation particle as a proclitic with a long Alif when used with verbs (ما instead of م). This removes ambiguity for cases like the local Arabic negated verb ماكان (there was not) from the MSA noun مكان (place) and the local Arabic هدا ما كان (that’s it). All negated verbs in local Arabic are spelled with ما as proclitic and ش as enclitic. As a result it is easier to get the non-negated form by stripping off the negation

[1] Berber has its own script called Tifinagh and a standardised Latin spelling.

clitics. By removing the initial ما and the final ش from ماعيطش (he did not call) we get عيط (he called).

3.2.6 Word segmentation and tokenisation

Users tend to spell prepositions, reduced question words and conjunctions as proclitics. This creates an unnecessary sparse and large vocabulary. To reduce the size of the vocabulary, we write such proclitics as separate full forms, among others: وش ، ح ، م ، في ، ولا ، ه ، عل ، ع ، تع ، بلا، ك ، غ ، شا ، واش ، ف ، او ، كي. We split لي and ما when they occur as relative pronouns attached to a verb. وشيكون (who is him) is tokenised to واش يكون, مالازمة (from the crisis) as من الازمة , حنديرلو (I will make him) راح نديرلو, and split relative pronouns in ماتتمناي (what you wish) as ما تتمناي and ليكان (who was) as كان لي. Other ambiguous cases include ورانا which could be either وين رانا (where are we) or و رانا (and we are) or وراءنا (behind us) depending on the context.

3.2.7 Abbreviations and acronyms

We collapse acronyms written as several tokens to a single token and extend abbreviated words to their full form based on their context. For instance, لاس أم اس is collapsed to لاساماس (SMS), and م ك is extended to مغرف كبيرة (tablespoon).

4 Data Statistics and Alignment

4.1 Data statistics

The final processed parallel corpus, described in Section 3 consists of 185,219 unique (input, output) text pairs where the input is from the automatically pre-processed data and the output is from the manually corrected and normalised data. Table 6.1 shows statistics about the parallel corpus where input corpus refers to the not corrected and normalised corpus and the output corpus refers to the corrected and normalised version. In more

details, 90.20% of types (unique words) in the input corpus occurred less than 10 times and 59.60% of them occurred only once in the entire corpus. These figures serve to give an idea about how sparse the data is. The difference in the vocabulary size between the two corpora (50,456 words and 26,199 types) is primarily because of the introduced transformations.

Corpus	# words	# types	longest text (# words)
Input	3,175,788	272,421	112
Output	3,125,332	246,222	112

Table 6.1 Statistics about the parallel corpus.

4.2 Data alignment

Another difference between the two corpora is that the lengths of the input and the output may vary as a result of different tokenisation. This is not a problem in terms of machine learning, because the models described in Section 5 are designed to deal with variable length input and output sequences. However, because our two sequences are from the same language with the same meaning (the only difference is in spelling) we expect that alignment at the token level will lead to improved performance (see Section 6.1).

To this end, we have developed an aligner whose task is to make sure that every single unit (token) in the input (with potential misspelling) matches a unit (token) in the output. This may seem trivial until one remembers that misspellings may include added or deleted spaces. Our aligner works by computing the minimal edits to transform the input into the output (using the standard Levenshtein distance algorithm).

These minimal edits are not the basis for training (they will be discarded) *unless* they concern spaces. If a space is added, then to preserve word alignment we replace the corresponding space in the output by a special symbol (#). In inference mode (see Section 5.4), this symbol will be replaced by a space. If on the contrary a space is deleted, then it is added back (and words are aligned again). A special extra symbol ($) is added to mark that a spurious space was added and should be eventually deleted again when the model is used in inference mode. This alignment algorithm provides correct results whenever the Levenshtein distance at the sequence level is the sum of the Levenshtein distances for each unit (token) that is misspellings are not so large as to make deleting/inserting whole words a shorter operation than changing characters within words; and this condition is satisfied in our corpus.

5 Models

We frame the task of spelling correction and normalisation as a sequence-to-sequence (seq2seq) prediction problem, i.e., given an input sequence what is the best predicted output sequence. Note that sequence refers to user texts of any length including one token or more. We use an encoder-decoder architecture which consists of two neural networks where the first one reads an input sequence and transforms it into a vector representation, and the second one, representing a language model, conditions the generation of output symbols on the input sequence representation and generates an output sequence (Cho et al., 2014a).

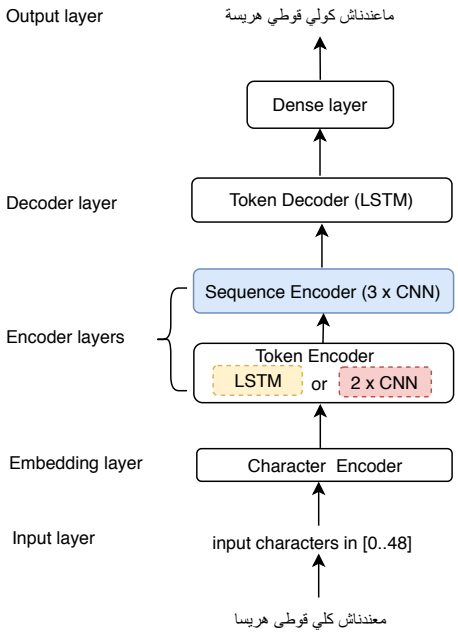


Figure 6.1 Model architecture.

As shown in Figure 6.1, the encoder consists of two sub-neural networks, namely token encoder and sequence encoder.

5.1 Token encoder

It reads the input sequence character by character and outputs a vector representation for each token in the sequence. Two configurations are used: either an LSTM encoder or a CNN encoder.

- **LSTM encoder:** represented in yellow and takes as input character embeddings with vocabulary size of 49, 100 dimensions, token representation size of 50 and a dropout rate of 30%.

- **CNN encoder:** represented in red and takes as input character embeddings. It is composed of 2 CNN layers with 50 filters of size 5, a ReLU activation, a dropout rate of 20% followed by max pooling in the temporal dimension.

5.2 Sequence encoder

Represented in blue and consists of 3 CNN layers with 200 filters for the two first layers and 100 for the third layer, all filters have size 3, ReLU activation and 0.05 dropout.

5.3 Token decoder

It is composed of one character-based LSTM layer with the same hyper-parameters as the LSTM encoder, followed by a dense layer.

5.4 Training and inference

All models are trained end-to-end to maximise the likelihood of the output sequence conditioned on the input sequence for 150 epochs using a batch size of 64 and Adam optimiser. Gradients with a norm greater than 5 are clipped.

For inference (generating an output character sequence), we use beam-search with a size of 20. Note that beam-search is used only to generate an output sequence and does not influence neither model training nor validation. The models generate characters starting from the start symbol (<) and stop at the end symbol (>) or at a predefined sequence length given as a hyper-parameter, whichever comes first.

6 Experiments and Results

In order to test our models and the gain from the aligner (see Section 4.2), we experiment with both versions of data: the non-aligned and the aligned data. It is worth mentioning that the only difference between them is that the aligned one contains extra symbols (# and $) marking missing or extra spaces. An extra space – thus word– is also added for every dollar sign. Moreover, to measure the effect of the context, we feed the data either token-by-token or sentence by sentence.

We split both the datasets into 75% (138,917 samples) for training, 5% (9,261 samples) for development, and 20% (37,041 samples) for validation. The reported hyper-parameters in Section 5 were fine-tuned on the development set.

We conduct two evaluations: (i) how well the suggested models perform on the seq2seq task, and (ii) how good is the best performing model for spelling correction and normalisation task, and what is its effect as a pre-processing step on downstream tasks like Semantic Textual Similarity (STS). We evaluate (i) using character-level accuracy, and we evaluate (ii) by calculating Precision, Recall and the F-score for the class of tokens that should be changed. Hence, Recall is the ratio of the correctly changed tokens to the number of tokens that should be changed, and Precision is the ratio of the correctly changed tokens to the number of tokens that are changed. F-score is the harmonic average of both.

	Models	Input	Data	Validation
1	LSTM-Token-seq	sequence of tokens	non-aligned	23.90
2	CNN-Token-seq	sequence of tokens	non-aligned	89.20
3	CNN-Token-seq-alig	sequence of tokens	aligned	**96.20**
4	CNN-Token-token-alig	one token	aligned	87.10

Table 6.2 Accuracy (%) of models on Seq2seq task.

6.1 Comparing models on Seq2seq task

In Table 6.2 we report the overall character level accuracy of the 4 best performing models for each configuration and experiment: (1) LSTM-Token-seq: the model with the Token LSTM + Sequence encoder (yellow and blue parts of Figure 6.1) and Token decoder, (2) CNN-Token-seq: the model with the Token CNN + Sequence encoder (red and blue parts of Figure 6.1) and Token decoder. Both (1) and (2) are trained and evaluated on non-aligned data with a sequence of tokens as input. (3) CNN-Token-seq-alig the same as model (2) but trained and evaluated on aligned data. (4) CNN-Token-token-alig: the same as (3) but with one token as input (token-by-token).

Results indicate that the LSTM encoder in (1) does not suit our task / data and fails to learn the sequential representations with an overall character accuracy of only 23.90%. This could be because of the high sparsity of the data which makes it hard to learn regularities. In contrast, the CNN encoder in (2) performs much better, with an overall character accuracy of 89.20%, suggesting that learning sequences of patterns through convolutions suits better our task / data than sequence modelling with LSTM. This is in line with what has been reported for machine translation in (Gehring et al., 2017).

The CNN encoder performs even better with the aligned data in (3). The difference can be attributed to the positive effect of the aligner which boosts the accuracy by 7%. The 9.1% drop in the accuracy in (4) compared to (3) is due to the lack of word context.

This indicates that word context is essential, especially for word sense disambiguation in such highly varied data.

6.2 Quality and effect

- **Quality** We use the best performing model (3) and run the inference mode, (see Section 5.4), on the validation set which contains 567,308 words of which 507,429 words are already correctly spelled and 59,880 words must be changed, either corrected or normalised. We perform quantitative and qualitative analysis of the generated sequences in terms of the changed spellings at a word level. Model (3) achieves an overall F-score of 64.74%, Recall of 88.58% and Precision of 51.02% on the words to change. It correctly spells 53,041 words from the total words to change and fails to correctly change 6,839 words. However, it introduces 50,914 incorrect changes (newly misspelled words or infelicitous corrections).

- **Error analysis** Examining the generated sequences shows that most errors are at the level of one character (duplicating or substituting one character) and the generated words are very similar to the reference. This is similar to the conclusion of Tiedemann (2009) that many errors of a character level phrase-based statistical machine translation for Norwegian and Swedish are of small length. Furthermore, we find that most of the not properly corrected words actually do not have enough representative instances, i.e., most of them occurred only once in the validation data and were not seen during the training. The high sparsity of the data is an interesting challenge for the current neural networks for which more research is needed.

  With the settings of our experiments, the high Recall of the model at a word level indicates that it can be used for detecting errors and words to normalise but not for *automatically* fixing them because of its low Precision. Actually the reported low Precision is not that dramatic as it might seem because it is aggressive, i.e., a single wrong character means the entire word is wrong. Besides improving our inference settings, a better metric for evaluating such cases is needed.

- **Effect** We evaluate the effect of spelling correction and normalisation, as a pre-processing step for downstream tasks, on detecting binary Semantic Textual Similarity. We chose this task because it is one of the few available tasks for ALG we are aware of. We apply our spelling correction and normalisation on the ALG data reported by Adouane et al. (2019). We replicate the best performing model for which the authors report an accuracy of 92.76%, and we get an accuracy of 94.40% with the same settings. The gain indicates that the spelling correction and normalisation is potentially a useful pre-processing step for downstream tasks.

7 Conclusions and Future Work

We compiled a new parallel corpus for ALG with linguistically motivated decisions for spelling correction and normalisation. Considerations such as being practical to implement and suitability for our goals are taken into account. We designed, implemented and tested 2 deep neural network architectures trained end-to-end to capture the knowledge encoded in the corrected and normalised corpus. The results showed that a CNN token-sequence encoder and an LSTM decoder performed the best when including context information. Additionally, applying a token aligner on the input data yielded better performance compared to the non-aligned data. Even though, with the current inference settings, the model generated some errors at a character level mainly due to the data sparsity, it is general and does not require extra resources except a parallel corpus. Hence it could be applied to other languages with the same settings.

In future work, we plan to improve the current inference mode by investigating other settings, improve the decoder by pre-training on the corrected and normalised data and a large MSA corpus to avoid generating incorrect character sequences. Moreover, we will evaluate the model extrinsically by using it to pre-process data for tasks such as code-switch detection, and topic detection to see how much it helps or hinders attempts to tackle these tasks.

Chapter 7

Identifying Sentiments in Algerian Code-switched User-generated Comments

Wafia Adouane, Samia Touileb, and Jean-Philippe Bernardy

Abstract

We present in this paper our work on Algerian language, an under-resourced North African colloquial Arabic variety, for which we built a comparably large corpus of more than 36,000 code-switched user-generated comments annotated for sentiments. We opted for this data domain because Algerian is a colloquial language with no existing freely available corpora. Moreover, we compiled sentiment lexicons of positive and negative unigrams and bigrams reflecting the code-switches present in the language. We compare the performance of four models on the task of identifying sentiments, and the results indicate that a CNN model trained end-to-end fits better our unedited code-switched and unbalanced data across the predefined sentiment classes. Additionally, injecting the lexicons as background knowledge to the model boosts its performance on the minority class with a gain of 10.54 points on the F-score. The results of our experiments can be used as a baseline for future research for Algerian sentiment analysis.

1 Introduction

Sentiment Analysis (SA) is a well-established NLP task which is commonly framed as a text classification problem. SA includes various related applications, depending on the domain and its real-world use, such as product reviewing and opinion mining. It has been applied to several domains, mostly for curated monolingual data. Recently, however, the focus has been extended to new domains and settings, namely to user-generated data which reflects real-world use cases.

In this paper, we focus on identifying and classifying sentiments by analysing user-generated comments on YouTube videos. We work with comments written in Algerian Arabic, a non-standardised Arabic variety characterised by heavy code-switching between co-existing languages and language varieties mainly Modern Standard Arabic (MSA), Berber, French, and local Arabic variants.

There is a large body of work on SA for Arabic, largely for MSA and Middle Eastern Arabic varieties (Rushdi-Saleh et al., 2011; Abdul-Mageed and Diab, 2012; Zbib et al., 2012; Aly and Atiya, 2013; Nabil et al., 2015; ElSahar and El-Beltagy, 2015; Salameh et al., 2015). Even so, there has been less work done for Northern African Arabic varieties, which are indeed colloquial languages, due primarily to the scarcity of written linguistic resources.

However, new social media platforms made it possible to obtain user-generated data reflecting real use of such languages. In interactive communication channels, users use speech-like languages to express themselves with spontaneous spelling for non-standardised languages. Hence this domain is potentially a useful resource for analysing the linguistic properties of this kind of unedited code-switched textual data, and therefore better understanding how these languages are naturally used.

This paper is an attempt to bridge the gap in sentiment analysis for user-generated data written in colloquial languages. As main contributions, (i) we introduce our newly built linguistic resources collected from YouTube (corpus and sentiment lexicons) for Algerian, and labelled for sentiments. To the best of our knowledge, this is the largest user-generated corpus labelled for sentiments for Algerian. (ii) We compare the performance of Support Vector Machine (SVM) and three end-to-end deep neural networks (DNNs) on identifying sentiments from code-switched colloquial language. (iii) We try to improve the best DNN models by injecting sentiment lexicons as background knowledge and by augmenting the number of instances for minority classes in training data.

In what follows, in Section 2 we briefly review related work. In Section 3 we describe our newly built linguistic resources, inter alia, corpus creation and labelling, sentiment lexicon compilation along with their detailed statistics. In Section 4 we present the approaches and the models that we use to identify sentiments from comments on social media. In Section 5 we describe our experiments and discuss the results. We conclude in Section 6 with our main findings and future plans.

2 Related Work

The largest body of research in sentiment analysis —including its various applications— is predominantly done for English. However, recently research has expended to other languages, such as Arabic and its colloquial varieties. Traditionally, sentiment analysis

was heavily based on sentiment lexicons (Turney, 2002; Hu and Liu, 2004; Taboada et al., 2011). That is to say, a sentiment of a text segment, be it a document or a sentence, was computed based on sentiment lexicon lookup by counting the number of positive or negative words. This approach, however, has proven to be limited because negation, intensification, downtoning, etc., can alter the polarity of words (Taboada et al., 2011). There have been a few attempts to overcome these limitations, among others, Taboada et al. (2011) who considered classifying the polarity of documents using lexicon information and rules for negation and intensification. They reported that their approach outperformed machine learning when tested across domains.

The focus in sentiment analysis has lately shifted to the use of end-to-end learning using information from the training corpus, and sometimes relying on the use of pre-trained embeddings and incorporating sentiment knowledge into their models. Moreover, Lei et al. (2018) used two sentiment lexicons, and Shin et al. (2017) used extra information from six lexicons with a Convolutional Neural Networks (CNN), and reported competitive results on one of the SemEval-2016 tasks. Similarly, Barnes et al. (2019) incorporated lexicon information into a Bidirectional Long Short-Term Memory (BiLSTM) sentiment classifier using a multi-task learning framework. Nonetheless, most approaches for sentiment analysis have not incorporated lexicons, but they heavily rely on end-to-end supervised approaches. For sentence-level sentiment analysis, Kim (2014) and Dahou et al. (2016) used CNN models, Qian et al. (2017) used a variation of LSTM, and Tai et al. (2015) used tree-structured LSTM.

Sentiment Analysis work for Northern African Arabic was done, among others, by Elouardighi et al. (2017) who used a combination of various features to train an SVM, random forests, and decision trees to classify MSA and Moroccan Facebook comments. At the same time, Medhaffar et al. (2017) used SVM, Binary Naive Bayes, and a Multi-Level Perceptron trained on three different corpora: an MSA corpus (OCA – (Rushdi-Saleh et al., 2011)), a corpus of MSA and other Arabic dialect (LABR – (Aly and Atiya, 2013)), and TSAC (Tunisian Sentiment Analysis Corpus) corpus, a code-switched corpus, to prove that it is necessary to train classifiers on dialects to achieve good accuracy. More recently, Jerbi et al. (2019) used the code-switched corpus TSAC presented in Medhaffar et al. (2017) for sentiment classification into the two classes positive and negative using the deep neural approaches LSTM, BiLSTM, deep-LSTM, and deep-BiLSTM, along with word embeddings. The authors reported that their approach of deep-LSTM outperformed the models presented by Medhaffar et al. (2017), and achieved and overall accuracy of 90%.

With respect to sentiment analysis for Algerian specifically, little work has been done. Mataoui et al. (2016) proposed a lexicon-based approach for SA on Facebook comments using resources translated from MSA to generate three lexicons (of keywords, negations,

and intensification words), and a list of emoticons and "common" Algerian phrases used to express polarity. They reported an accuracy of 79.13%. Then Guellil et al. (2018) automatically translated an English sentiment lexicon to Algerian while keeping and transferring the polarity of the English words. They also automatically labelled Facebook comments as positive or negative based on the constituents' polarities, using a Bag-of-Words (BoW) model and document embeddings. The authors used five different sentiment classifiers, namely SVM, Naive Bayes, Logistic Regression, Decision Trees, and Random forest and reported an F1-score of 72 for comments written solely in Arabic script, and 78 for comments written in Latin script both achieved using Logistic Regression. Likewise Soumeur et al. (2018) manually labelled a corpus of more than 25,000 comments collected from Facebook pages of 20 companies into positive, negative, and neutral classes. However, they translated all code-switched segments into Arabic words and transliterated words written in Latin script into Arabic script. They reported that a CNN model with a BoW representation achieved the best performance with an accuracy of 89.5%.

Unlike in the earlier mentioned work, in all our resources in this work, we use user-generated data without any transformation except for a simple automatic pre-processing done to reduce the size of the vocabulary. Furthermore, we take advantage of deep neural networks trained end-to-end to identify sentiments from unedited and code-switched user-generated comments written in Algerian in both Arabic and Latin scripts. Also, we experiment with two ways to improve the performance of our models, notably injecting the sentiment lexicons as background knowledge to a CNN model and augmenting the training data to overcome the problem of unbalanced data.

3 Linguistic Resources

3.1 Corpus creation and properties

To the best of our knowledge, there are no adequate Algerian corpora labelled for sentiment analysis that would serve our purpose, we therefore created our own corpus. To do so, we compiled a list of 139 popular Algerian YouTube channels that span a wide range of genres from news, politics, sports, cooking, vlogs, product reviews, and TV-shows. We manually collected 50,000 comments or posts of different lengths, and removed all comments that were not written in Algerian, such as those written in Modern Standard Arabic (MSA) or another Arabic dialect, namely Moroccan, Tunisian, Egyptian, etc. We also removed all comments that were entirely written in Latin script (French or Arabic written in Latin script), but we kept those written in mixed scripts –Arabic and Latin. Likewise, we kept all comments that were a mix of MSA, Berber, French and local Algerian Arabic. This decision is based on the fact that Algerian Arabic is a mixture of co-existing

languages and language varieties written in non-standardised orthography both in Latin and Arabic scripts.

The example in (1) is a user-generated comment written entirely in Arabic script which displays code-switching at a word level between several varieties; exhibited in the use of the following words: MSA (عندك، الحق، عندي، و، عام، ما، حتى، مشكل – you have, right, I have, and, year, not, any, problem), French (لاماشين – washer), and local Algerian Arabic: (هاذي، ختها، وبزاف، مليحة، ملي، شريتها، معاها – this, like it, very, good, since, bought it, with it). Note that in *formal* MSA people might express the same meaning using other words.

(1) a. عندك الحق هاذي لاماشين عندي ختها وبزاف مليحة وعندي عام ملي شريتها وما عندي معاها حتى مشكل

b. You're right, I have the same washer and it is very good. I have had it for a year and have had no problems with it.

(2) a. لازم نبكو مع بعض ونفرحو مع بعض مالا حنا علاه نتبعو فيك غير لا ضحك نن حبيبتي نحبوك وكلش فيك نحبوه **bn courage bn continuation**.

b. We should support each other otherwise why are we following you, just to laugh! No dear, we love you and love everything in you, good luck!

The example in (2) also displays code-switching, between MSA (مع، بعض، ضحك، حبيبتي، فيك – with, each other, dear, in you), French (نن – no) and local Algerian Arabic (لازم، نبكو، ونفرحو، مالا، حنا، علاه، نتبعو، غير، لا، نحبوك، وكلش، نحبوه – we must, we cry, we will be happy, so, we, why, follow, just, no, we love you, and everything, we love it). In addition to the code-switching at a word level, the user mixes Arabic and Latin scripts. Because local Algerian Arabic is a colloquial language with no standardised orthography, people use speech-based spelling particularly lots of spelling variations reflecting local/regional pronunciation. We leave the data unedited, with typos and misspelling of MSA words.

We applied the following cleaning procedure on the raw data. The encoding of Arabic characters was normalised so that equivalent characters were mapped to a single Unicode point. All comments were anonymised, i.e. users' information was deleted manually and all mentions of people were automatically replaced by others, while keeping their context meaningful. Mentions of celebrities and political figures were kept, generic references,

such as ختي (sister), خو، خويا (brother), and حنونة، حبيبتي، حبيبنا، شريكي، صاحبي (friend) were also kept. Long comments were trimmed and split where there was a clear split, both at sentence boundaries and when a user clearly expressed opinions on two different topics.

As for its statistical properties, after cleaning, our corpus consists of 36,120 unedited short colloquial comments or relatively short texts with an average length of 14.47 tokens or 74 characters, as is expected given the data source (social media). It comprises 522,890 tokens (lexical words, digits and emoticons, punctuation not included) and 78,482 unique tokens.

The corpus displays also the linguistic properties of Algerian Arabic, mentioned earlier, namely code-switching, spelling variations and spelling errors for MSA, hence increasing the data sparsity. Another property of our corpus is that it consists of discussions and sub-discussions—in essence short written multilogues. We kept the order of the comments exactly as found on the platform (YouTube) to keep a larger context. That is, a user may refer to different things at the same time in one short comment, such as comment on a video, comment on previous comments, and talk about personal experiences or something completely unrelated. Users also quote each other a lot, or quote segments from the video or from previous comments and comment on them.

3.2 Corpus labelling and statistics

In sentiment analysis, texts are commonly classified based on their polarity: either as positive (POS) or negative (NEG), or neutral (NEU). In our case, users comment on videos or give their feedback on something related or unrelated, they agree, disagree, give their own experience, or add new information. Therefore to be realistic and model the user-generated data at hand, we decided to add the class MIX for the cases where users combine polarity or add new information in addition of the three standard classes (POS, NEG, NEU).

More precisely, our labelling guidelines are as follows: use POS or NEG if it is understood from the comment that its generator is clearly expressing something positive or negative as in examples (3) and (4) respectively; use NEU if the comment does not bear any sentiment, like inserting a piece of information or a quote, or asking questions as in example (5); otherwise use MIX for cases combining POS, NEG and/or NEU as in example (6) where the user is positive about the video presenter and negative about previous comments. Two Algerian native speakers labelled the corpus described in Section 3.1, taking the users' perspectives into account.

(3) a. هه نموت عليك عندك عقلية فوورر

b. I love you I love your way of thinking.

(4) a. نيفو زيرو وش هاد تمسخير تزيدو توريوه

b. Low level what is this joke and you dare to show it!

(5) a. شحال فعمرك كي بديت

b. How old were you when you started?

(6) a. باين غيارين برك ياك يكرهو لي ناجح الله لا تربحهم انت طوب كملي

b. It is clear they are just jealous (of you) they hate someone who succeeds may God fail them, you are top carry on!

The final labelled corpus does not maintain balance between classes. Precisely, as shown in Table 7.1 below, we have 10,698 comments for POS, 6,424 for NEG, 11,736 for MIX, and 7,262 for NEU.

	POS	NEG	MIX	NEU
Train	7,330	5,017	8,357	5,682
Dev	1,285	559	1,187	581
Test	2,083	848	2,192	999
Total	10,698	6,424	11,736	7,262

Table 7.1 Distribution of comments over classes.

We measure the reliability of the human judgments by computing the inter-annotator agreement (IAA), corresponding to how often annotators made the same decisions (agreed) and how many times they made different decisions (disagreed). To this end, we shuffled the data and randomly selected 2,000 comments (labelled by the two annotators of the entire corpus). Their IAA computed using standard *Cohen's kappa coefficient* κ is 0.75.

To gain more confidence on agreement, we also asked two additional Algerian native speakers to label the same 2,000 comments following the same guidelines as above.

We calculated the inter-annotator agreement with four annotators: the two annotators of the entire data plus the two additional ones who were separately asked to label the previously mentioned sample of 2,000 comments. All annotators worked independently from each other after agreeing on labelling guidelines. We used two IAA measures: *Krippendorff's alpha* and *Fleiss' kappa*, and obtained the following scores: *Krippendorff's alpha*

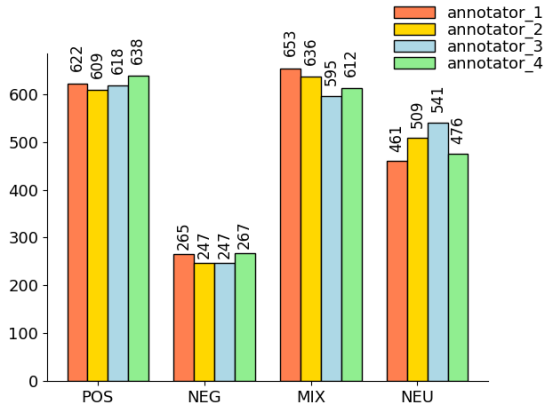


Figure 7.1 Per-class overview of the agreements between four native annotators.

α = 0.89, and *Fleiss' kappa* κ = 0.89. These values provide strong evidence for good agreement. The per-class tallies shown in Figure 7.1 indicate that annotators disagreed more on the MIX and NEU classes.

3.3 Sentiment lexicons

We constructed small positive and negative sentiment lexicons using the entire labelled corpus. To do so, we first identified the 2,000 terms that are the most correlated with each of the positive (POS) and the negative (NEG) classes using the chi-squared test. We looked for both unigrams and bigrams. We then curated these lists, and manually added a list of most common Algerian positive and negative words and emoticons that were not already in the lists. This resulted in a positive lexicon of 917 entries and a negative lexicon containing 647 entries. The lexicons reflect the nature of the Algerian language and contain both borrowings and code-switched entries.

4 Models

We frame the task of identifying sentiments from social media as a text classification problem. That is, given a comment (text segment of any length) predict the sentiment among the predefined classes. We compare the performance of four models: (1) a Linear Support Vector Machine (SVM) with bag-of-words representations of the data as features. Three Deep neural models with different configurations, (2) using Convolution Neural Network (CNN), (3) using Long Short-Term Memory (LSTM), and (4) using Bi-directional Long Short-Term Memory (BiLSTM) as summarised in Figure 7.2.

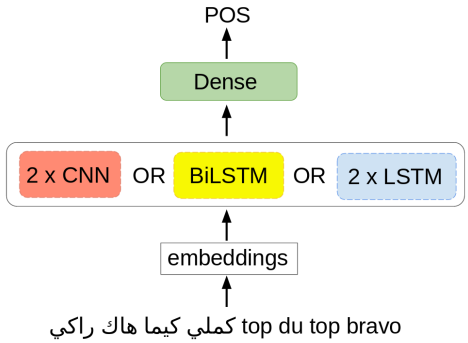


Figure 7.2 Model architectures. Each of the 3 DNN models takes a user comment as input and outputs a class from the set of sentiment classes POS, NEG, NEU, and MIX.

- **CNN** (in dark orange), is comprised of two passes, one creating word representations from characters, and one taking a sequence of these representations and classifying it. To construct word representations, we first use a character embedding layer mapping each of the 430 possible characters to a 50-dimensional representation. Then we use a convolution layer with filter size 3 followed by ReLU activation and max-pooling in the time domain. The sentence-level analysis is composed of two convolution layers. The first layer has 50 features and the second has 30. Both layers use a filter size of 3 with a dropout rate of 15%, followed by ReLU activation. This architecture is similar to that proposed by Adouane et al. (2019), but uses different hyper-parameters.

- **BiLSTM** (in yellow) takes word embeddings with a vocabulary size of 326,847 and embeddings dimension of 100. It consists of one BiLSTM layer with 100 units with a dropout rate of 10% followed by a global max pooling layer.

- **LSTM** (in blue) takes as input the same word embeddings as in the BiLSTM. It is composed of 2 LSTM layers where the first layer has 200 units and the second has 100 units with dropout rate of 20% between the layers.

In each of the three configurations (CNN, BiLSTM and LSTM), the final stage of the DNN is a dense layer (in green) with a softmax activation layer which maps their outputs to sentiment classes. All neural models are trained end-to-end for 60 epochs using a batch size of 64 for CNN and 128 for LSTM and BiLSTM, and Adam optimiser. Gradients with a norm greater than 5 are clipped.

5 Experiments and Results

In order to evaluate the performance of the models on identifying sentiments, we shuffle the data and randomly pick 6,122 samples as a test set, 3,612 samples as a development set and the remaining 26,386 as a train set.

The hyper-parameters mentioned in Section 4 were fine-tuned on the development set. Table 7.1, mentioned earlier, shows the number of comments for each set by sentiment class. Table 7.2 gives a summary of the number of tokens and comments for each set. Note that overlapping #Unique tokens between sets were counted only once, hence the total is less than the sum of the #Unique tokens of all sets.

	Train	Dev	Test	Total
#Comments	26,386	3,612	6,122	36,120
#Tokens	367,483	57,874	97,533	522,890
#Unique tokens	63,117	16,664	24,274	78,482

Table 7.2 Corpus statistics with distribution over the 3 sets. Note that overlapping #Unique tokens between sets were counted only once, hence the total is less than the sum of the #Unique tokens of all sets.

As shown in Table 7.3, in terms of overall accuracy, the BiLSTM outperforms other models with an accuracy of 66.78%, compared to LSTM with 66.67%, CNN with 65.37% and SVM with 61.65%. In terms of macro F1, our CNN is the best performing model. Nevertheless, because the data is unbalanced, we are more interested in the performance of the models for each sentiment class.

Model	Accuracy (%)	Macro F1
BiLSTM	**66.78**	48.53
LSTM	66.67	48.70
CNN	65.37	**60.17**
SVM	61.65	58.50

Table 7.3 Overall accuracy (%) and macro F1 of the models.

As you can see in Figure 7.3a, BiLSTM (in yellow) and LSTM (in blue) are too biased to the two majority classes. They perform well on POS and MIX classes achieving a F-score of 95.77 and 95.18 respectively on POS, and 93.46 and 93.13 on MIX. However, they perform poorly on the two minority classes, namely NEG and NEU with an F-score

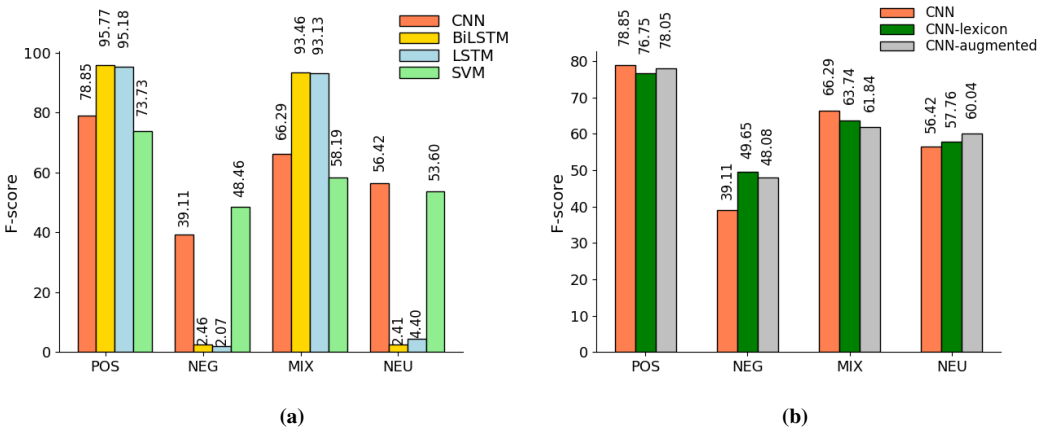


Figure 7.3 F-score of each model per sentiment class.

of only 2.46 and 2.07 on NEG, and 2.41 and 4.40 on NEU.

At the same time, the CNN (in orange) performs worse than BiLSTM and LSTM with an F-score of 78.85 on POS and even less on MIX achieving only 66.29 F-score. But it performs much better on NEG and NEU with an F-score of 39.11 and 56.42. Surprisingly the SVM (in light green) performs the best for NEG achieving an F-score of 48.46, and less than the CNN on NEU with 53.60 F-score. Still the SVM performs the worst on POS and MIX with only 73.73 and 58.19 F-score respectively.

Model	POS	NEG	MIX	NEU
BiLSTM	**95.86**	2.24	**93.39**	2.67
LSTM	95.64	1.92	92.84	4.75
SVM	75.59	**55.47**	52.91	**57.16**
CNN	78.16	40.87	65.97	56.08
CNN-lexicon	73.65	53.47	66.37	55.20
CNN-augmented	75.53	45.61	69.72	54.95

Table 7.4 Precision (%) of each model per sentiment class.

With regards to Precision and Recall, as shown in Tables 7.4 and 7.5 the CNN achieves a reasonably good Precision and Recall on all classes along with the SVM in comparison to the LSTM and BiLSTM.

Summing up the results, the CNN outperforms LSTM-based models and SVM for all classes except for NEG where it is outperformed by the SVM. This could be explained

Model	POS	NEG	MIX	NEU
BiLSTM	**95.68**	2.71	**93.52**	2.20
LSTM	94.72	2.24	93.43	4.10
SVM	71.95	43.02	64.65	50.47
CNN	79.55	37.50	66.61	56.76
CNN-lexicon	80.12	46.34	63.74	57.76
CNN-augmented	80.75	**50.83**	55.57	**66.17**

Table 7.5 Recall (%) of each model per sentiment class.

by the fact that deep neural models —based on stacking layers with many non-linear transformations— perform better with more data as they are able to learn increasingly more abstract representations. That is, learning the underlying hierarchical structures from the data (with CNN) better fits the data at hand compared to modeling its structures sequentially (LSTM). In fact, the CNN is the only tested model that accesses information at a character level, thus potentially making it robust to data sparsity in the form of misspellings, etc.

On the other hand, the Linear SVM —a binary classifier in its core— can not handle large unbalanced data with multiple classes. Unexpectedly it outperforms CNN for NEG class both in Precision and Recall (see Appendix). One possible explanation is that the number of NEG samples is not enough for the CNN to learn patterns from the noisy sparse data, and hence it is hard to extract useful features from the train data and generalise them to the test data.

In the following we experiment with other ways to improve the performance of the CNN model.

5.1 Injecting the sentiment lexicons to the CNN

One way to improve the CNN performance is to add information from the sentiment lexicons, described in Section 3.3, as background knowledge. To this end, we encoded the lexicons and injected them to our CNN model. As show in Figure 7.3b, the lexicons boosted the performance of the CNN (referred to as CNN-lexicon in dark green) by 10.54 F-score points gain on NEG and 1.34 on NEU. Nonetheless, its F-score dropped by 2.1 points on POS and 2.55 on MIX.

5.2 Adding augmented data to the CNN

To deal with the unbalanced data, we experimented with augmenting the number of minority classes by duplicating all NEG and NEU samples in the training set without changing the test set. We retrained the CNN-lexicon, see Section 5.1, with the augmented data and

referred to it as CNN-augmented. The results in Figure 7.3b (in grey) indicate that the data augmentation has a positive effect on NEU with a gain of 2.28 points in F-score mainly due to boosting its Recall with 8.41 percentage points.

On the other hand, its Precision on MIX increased while its Recall dropped by 8.17 points (see Appendix). The same trend is observed for NEG where Recall increased by 4.49 points, achieving the best results, Precision dropped by 5.22 points. On POS, the augmented data has a slightly positive effect both in terms of Precision and Recall. Overall, there is a trade-off between the Recall and Precision of the sentiment classes.

It is worth mentioning that we also experimented with pretrained embeddings as opposed to learning embeddings along with a model itself (not included). For this we trained word embeddings using FastText (Joulin et al., 2016) on a large user-generated data (Adouane et al., 2019) and plugged them into the deep neural models in Section 4. Nevertheless, the addition was not helpful, *i.e.* the difference was not clear compared to the models without the pretrained embeddings.

5.3 Error analysis

Looking to the confusion matrices (not shown for concision), we found that a common confusion of the CNN models with different setups (CNN, CNN-lexicon and CNN-aug) is between MIX and POS with 349, 424 and 440 cases respectively. Whereas LSTM and BiLSTM confuse mostly between NEU and NEG with 908 and 930 cases. This is reflected in the above reported results. Confusing MIX and POS could be related to the fact that the MIX class is defined as any combination of the rest of the classes, namely NEG, NEU and POS.

(7) a. نتي هايلا وعقليتك روعة فيك ديفو تعاودي بزاف لهدرة

b. You are gorgeous and your way of thinking is great, your flaw is that you repeat yourself a lot

(8) a. راكي فوور وموتو يا العديان كنترا فيكم

b. You are great, die you enemies, we are against you

(9) a. ديري ريجيم تنقصي تبهاي شوية

b. Follow a diet to loose some weight and you'll become a bit gorgeous

(10) a. فيديو ختامو مسك عيفتيلي قلبي

b. What a great video ending, disgusting!

(11) a. وشحال عجبتني تاع اكرهكي كرها شديدا

b. I love (when you said) I hate you so much.

We will discuss in what follows some of the misclassified examples by the CNN-lexicon model. The example in (7) is classified as POS instead of MIX. One potential solution to overcome this issue is to do a fine-grained sentiment identification at a segment level instead of comment-level.

Also it is not easy to identify the NEU class. The reason is that there are lots of irony, humour, sarcasm, and metaphors in our corpus. This means that the sentiment is sometimes not conveyed by the literal meaning of words, which are *neutral* when taken individually, but rather need a pragmatic interpretation which is only accessible when taking into account the larger cultural context. Additionally, the class can depend on the perspective taken (that is to say commenting on the content of the video, or on previous comments, or on users' (un)related personal experience). The example in (8) is misclassified as POS while human annotators classified it as MIX taking into account the user's perspective where s/he likes the video and is against the previous comments criticising it.

The example in (9) is classified as NEU by the CNN (it does not have an explicit sentiment) and MIX by human annotators with the interpretation of implicit NEG + advice. Whereas the example in (10) is NEG (with irony) but the CNN classified it as POS.

Moreover, as mentioned earlier, there are many instances of quoting, like referring to something in a video or mentioning something said before. Such cases could be interpreted either as POS by someone who did not see the video or read previous comments (lack of context) or as NEU by someone who knew the context. For instance, the user in example (11) liked that the video presenter said that 'she hates someone so much'. But the quote itself is a negative statement attributed to the video presenter, and thus the example is classified as NEG by the CNN.

Analysing the confusions of the CNN-lexicon shows that some of them are in line

with the inter-annotator agreement in Figure 7.1 where the variation is more on the MIX and NEU classes.

6 Conclusions and Future Work

We presented in this work our new manually built linguistic resources for Algerian: a corpus which consists of more than 36,000 user-generated comments labelled for sentiments, along with sentiment lexicons of positive and negative words. We then described our models utilised to automatically identify sentiments from the user-generated comments. We discussed the performance of each model per sentiment class measured as F-score.

We found that the CNN model trained end-to-end fits better our data across the predefined sentiment classes compared to SVM, BiLSTM and LSTM models. Adding the lexicons as background knowledge boosted the performance of the CNN even more on minority classes (NEG and NEU) to different extents. Moreover, analysing the confusion matrix showed that it is quite challenging to distinguish between MIX and NEU classes.

As future improvements, we plan to do fine-grained classification on the MIX class, i.e. identify sentiments at the segment level instead of at the comment-level. A difficulty to overcome is that a comment may be globally negative while a large segment can still be positive, as in example (10). Also being able to identify where in a comment a user has switched from being negative to positive (or vice versa) can be challenging.

Additionally, we will experiment further with multitask learning for which we have preliminary results (not in the scope of this paper). For this, we plan to (i) jointly learn sentiment classification as main task and lexicon prediction as auxiliary task as presented in Barnes et al. (2019), and (ii) train our best performing model jointly with other tasks and investigate whether and where there will be any performance gain.

Chapter 8

When is Multi-task Learning Beneficial for Low-Resource Noisy Code-switched User-generated Algerian Texts?

Wafia Adouane and Jean-Philippe Bernardy

Abstract

We investigate when is it beneficial to simultaneously learn representations for several tasks, in low-resource settings. For this, we work with noisy user-generated texts in Algerian, a low-resource non-standardised Arabic variety. That is, to mitigate the problem of the data scarcity, we experiment with jointly learning progressively 4 tasks, namely code-switch detection, named entity recognition, spell normalisation and correction, and identifying users' sentiments. The selection of these tasks is motivated by the lack of labelled data for automatic morpho-syntactic or semantic sequence-tagging tasks for Algerian, in contrast to the case of much multi-task learning for NLP. Our empirical results show that multi-task learning is beneficial for some tasks in particular settings, and that the effect of each task on another, the order of the tasks, and the size of the training data of the task with more data do matter. Moreover, the data augmentation that we performed with no external resources has been shown to be beneficial for certain tasks.

1 Introduction

New breakthrough results are continuously achieved for various natural language processing (NLP) tasks, often thanks to the availability of more data and computational power. Likewise, various learning frameworks have been proposed for NLP including multi-task learning. Multi-task learning is about transferring knowledge learned in one task to other tasks by sharing representations (Caruana, 1997). The assumption is that the final learned

shared representations are conditioned on the multiple tasks learned simultaneously, and as such they generalise better compared to separate training for each task. This works well when the jointly learned tasks are beneficial for each other, or in cases where a well-performing (auxiliary) task with large data is trained with a related (target) task with less data. However, predicting when tasks are useful for each other remains an open theoretical question and the reported results are still experimental.

This paper is an attempt to take advantage of the state-of-the-art advances in NLP, namely deep neural networks (DNNs) and multi-task learning in order to mitigate the problem of the scarcity of labelled data for colloquial Algerian language (henceforth referred to as ALG). Our main contributions are (1) the creation of a new dataset for code-switched Named Entity Recognition for ALG. (2) An investigation of the settings where it is beneficial to share representations to transfer the knowledge learned in one task to another or to other tasks. To this end, we jointly train 4 tasks: (1) Code-Switch Detection (CSD), (2) Named Entity Recognition (NER) —both framed as sequence tagging— (3) Spelling Normalisation and Correction (SPELL) —framed as a sequence-to-sequence task— and (4) identifying users' sentiments (SA) —framed as a classification task.

We analyse (1) the effect of each task on another, (2) whether task order matters or not, (3) whether word context for the sequence-to-sequence task is important or not, (4) whether the size of the training data of the task with more data matters, and (5) whether it is useful to augment the training dataset of sequence-to-sequence task (this does not require any extra resources). We believe that this investigation will extend the utility of multi-task learning in low-resource settings. In our experiments we increase the difficulty of the tasks gradually, for instance learning the tasks in pairs, 3 tasks, then 4 tasks, and increase the size of the training data for SPELL progressively.

The paper is organised as follows. In Section 2 we review related work. In Section 3 we describe our tasks and their corresponding datasets. In Section 4 we present the architecture of our model. In Section 5 we describe our experiments and discuss the results. In Section 6 we conclude with the main findings and outline potential directions for future improvements.

2 Related Work

In general, the definition of a task is vague and it could refer to an NLP task (Martínez Alonso and Plank, 2017), to a domain (Peng and Dredze, 2017) or to a dataset (Bollmann et al., 2018). Multi-task learning has been applied successfully to a variety of NLP tasks[1] (Collobert and Weston, 2008; Luong et al., 2016; Martínez Alonso and Plank, 2017; Bin-

[1] We cite here only a few examples.

gel and Søgaard, 2017), focusing on examining the effect of different auxiliary tasks on the performance of a target task. Changpinyo et al. (2018) use joint learning of 11 sequence tagging tasks, investigating whether doing so benefits all of the tasks. Based on the previously reported results, multi-task learning is a promising framework to improve learning with scarce data. Nevertheless, previous work has been mostly limited to morpho-syntactic and semantic sequence labeling tasks, *inter alia*, part-of-speech tagging, syntactic chunking, supersense tagging, semantic trait tagging, semantic role labeling, semantically related words, as well as multi-perspective question answering, and named entity recognition.

But what about the languages (domains) for which we do not have labelled data for morpho-syntactic and semantic tasks? Unfortunately many languages (or domains like user-generated data) do not have labelled data to perform such tasks. Indeed, NLP research is still focused largely only on a few well-resourced languages, and models are trained primarily on large well-edited standardised monolingual corpora, mainly for historical reasons or for current incentives. Additionally, in many cases the developed techniques fail to generalise (Hovy and Spruit, 2016), even to new domains within a single language (Jørgensen et al., 2015), mostly because they are designed to deal with a particularly structured corpora.

Accordingly, it is not clear whether the previously reported results using multi-task learning for NLP generalise to low-resource settings. In this work, we begin to answer this question by applying multi-task learning to user-generated data. As a case study, we take the language used in ALG which uses code-switching, non-standardised orthography as well as it suffers from the lack of any NLP tools such as a tokeniser or morpho-syntactic parsers. Like Changpinyo et al. (2018), we examine the settings in which our tasks benefit from multi-task learning, including pairwise tasks, order of the tasks and the size of the training data for the task with more data.

3 Tasks and Datasets

3.1 Tasks

In multilingual societies people have access to many linguistic codes at the same time. In diglossic situations people have access to even different linguistic levels of the same language (Major, 2002). It is the case in North Africa, where for historical reasons many languages and language varieties are used simultaneously at various extents, including mostly Berber, Arabic and French (Sayahi, 2014). These languages and language varieties coexist throughout the region and they are actively used on a daily basis (Rickford, 1990). Consequently in speech-like communications, such as in social media, people tend to mix

languages.

- **CSD** as mentioned in Chapters 2, 3 and 4, the task deals with the detection of the language (in multilingual CSD) or language variety (in diglossic CSD) of each word in its context for disambiguation. This is challenging for ALG, because the same script is used for all languages (MSA, local Arabic varieties, Berber, French, and English). To further complicate matters, vowels are omitted from the text.

  On the other hand, the enormous spelling variations in user-generated data for all languages and language varieties (Eisenstein, 2013; Doyle, 2014; Jørgensen et al., 2015) challenges the standard language ideology wrt. whether human languages are universally standardised and uniform (Milroy, 2001). It also poses serious challenges to the current NLP approaches at all linguistic levels.

- **SPELL** as mentioned in Chapter 6, the task aims at reducing orthographic variation and noise in the data, by context-dependent spelling correction and normalisation. Indeed, user-generated content in colloquial languages contains lots of spelling variations for these languages do not have standardised orthography and the content is unedited. We stress that SPELL is different from well-established spelling error correction task in that it deals with a non-standardised code-switched language —with no reference spelling.

- **SA** as mentioned in Chapter 7, the task deals with identifying users' sentiments from their generated comments.

- **NER** the task deals with the detection and classification of mentions referring to entities into pre-defined classes (person, location, organisation, product, company, etc.).

3.2 Datasets

For each task we use a separate labelled dataset. Table 8.1 shows statistics about the CSD, SA and NER datasets.

- **CSD** we use the dataset described in Chapter 2 and which consists of 10,590 user-generated texts labelled at a token level, and includes 9 classes, namely Local Algerian Arabic, Berber, French, English, Modern Standard Arabic, and Borrowing (which refers to foreign words adapted to the Algerian Arabic morphology), Named Entity as a general class, Interjections/sounds and Digits.

- **SPELL** we use the dataset described in Chapter 6 and which consists of a parallel corpus with 50,456 words and 26,199 types to be corrected or normalised.

CSD		SA		NER	
Class	Total	Class	Total	Class	Total
ALG	118,942	MIX	11,736	OOO	67,7191
MSA	82,114	POS	10,698	PER	7,262
FRC	6,045	NEU	7,262	LOC	4,641
BOR	4,025	NEG	6,424	PRO	3,682
NER	2,283			OTH	901
DIG	1,394			ORG	399
SND	687			COM	248
ENG	254				
BER	99				

Table 8.1 Statistics about the datasets: CSD (#tokens), SA (#samples) and NER (#mentions).

- **SA** we use the dataset described in Chapter 7 and which consists of 36,120 user-generated comments.
- **NER** we could not get any dataset labelled for NER for ALG that would serve directly our purpose. Therefore we compiled a new dataset by combining the two datasets used for CSD and SA, resulting in 46,710 user-generated comments in total. Then with the help of two other native speakers, we manually labelled it for NER by classifying every named entity mention in one of the 6 pre-defined classes, following the labelling schema used in OntoNotes Release 5.0 [2]. The classes are: person (PER), location (LOC), product (PRO), organisation (ORG) and company (COM). We tagged the rest of named entity mentions like time and events as "other" (OTH) to distinguish them from non-named entities (OOO). In order to identify multi-word expressions as one named entity chunk, we use the IOB (Inside-Outside-Beginning) labelling scheme. The newly labelled corpus for NER has 17,133 named entities with the IOB details.

4 Models

4.1 CSD and NER

We frame CSD and NER as sequence tagging tasks, i.e., the task is to assign one of the pre-defined tags to each token in an input sequence. We use a similar model architecture as the one described in Chapter 4. However, here the encoders are shared between the tasks, while the decoders are task-specific.

- The **Token-level encoder** (in dark orange in Figure 8.1) encodes the input sequence at the token level. It maps each of 430 possible characters (including special characters

[2] https://catalog.ldc.upenn.edu/docs/LDC2013T19/OntoNotes-Release-5.0.pdf

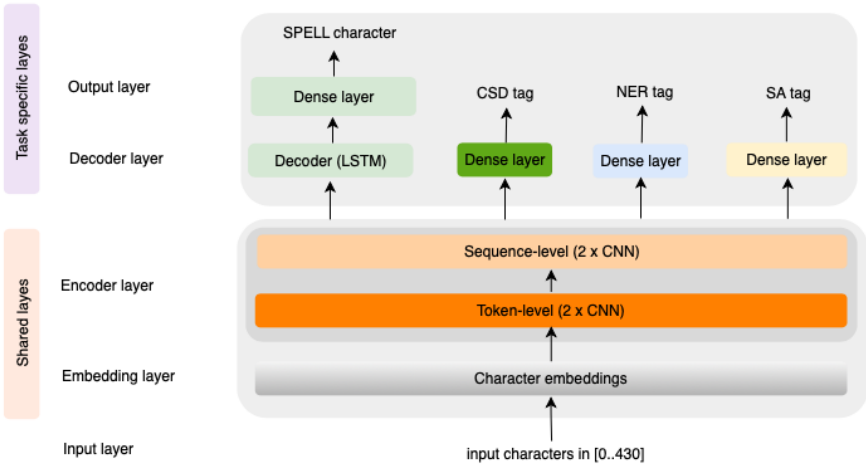


Figure 8.1 Multi-task model architecture.

and emoticons) to a 100-dimensional representation. It is composed of two convolution layers with 100 features and a filter size of 5 with a dropout rate of 20%, followed by ReLU activation and max pooling in the temporal dimension. In sum, it reads an input sequence character by character and outputs character embeddings for each token (constructs token representations).

- The **Sequence-level encoder** (in light orange) acts at a sequence level. It takes the outputs of the token-level encoder (character embeddings) and outputs word embeddings as a representation for the entire sequence. It consists of two convolution layers with 200 features for the first and 100 for the second, a filter size of 3, ReLU activation and a dropout rate of 5%.

- The **Dense layer** (in dark green for CSD and light blue for NER) with softmax activation maps the output of the sequence-level encoder (word embeddings) to CSD or NER tag sets respectively.

4.2 SPELL

We frame SPELL as a sequence-to-sequence prediction task where the input is a user-generated sequence (text) and the output is its normalised and corrected version (sequence). For this, we use an encoder-decoder architecture (Cho et al., 2014a) similar to the one described in Chapter 6.

- The **Encoder** consists of the shared layers described above.

- The **Decoder** (in light green) and consists of one Long Short-Term Memory (LSTM) layer (Hochreiter and Schmidhuber, 1997). It takes the output of the sequence-level encoder (word embeddings) as input and reads it character by character with a vocabulary size of 430, 100 units, a token representation size of 100 and a dropout rate of 10%. It is followed by a dense layer (in light green too).

4.3 SA

We frame SA as a text classification task: i.e., assign one of the pre-defined tag sets to an input sequence of any length. We use the model described in Chapter 7 which consists of two sub-neural networks.

- The **Encoder** consists of the shared layers described earlier, namely the Token-level and the Sequence-level encoders.

- The **Dense layer** (in yellow) with softmax activation maps the output of the sequence-level encoder to SA tags.

All models are trained end-to-end for 50 epochs using a batch size of 64 and Adam optimiser. Gradients with a norm greater than 5 are clipped. As the main focus of the multi-task learning, models share embedding and encoder parameters. Each task is run for a full epoch before switching to the next task. Therefore there is no special code to combine losses (each loss function remains the same for a whole epoch).

5 Experiments and Results

In order to evaluate the performance of our model, we shuffled the datasets and split them (with no overlapping parts) as follows.
For **CSD** we use 30% (3,177 samples) as a test set, 10% (1,059 samples) as a development set, and the remaining 60% (6,354 samples) as a training set.
For **SPELL** we use 20% (37,041 samples) as a test set, 5% (9,261 samples) as a development set, and 75% (138,917 samples) as a training set.
For **SA** we use 17% (6,122 samples) as a test set, 10% (3,612 samples) as a development set, and 73% (26,386 samples) as a training set.
For **NER** we use 30% (14,013 samples) as a test set, 10% (4,671 samples) as a development set, and the remaining 60% (28,026 samples) as a training set.

Note that all datasets are separate and are labelled for different tasks using different tag sets (depending on the task). The hyper-parameters mentioned in Section 4 are fine-tuned on the development sets. Given the small size of the CSD dataset and the high sparsity

of the SPELL dataset, after fixing the hyper-parameters, we train both on the training and the development sets, following Yin et al. (2015).

To examine the effect of jointly learning the tasks, we experiment with the following setups:

1. **Pairwise tasks**: To measure the effect of a task on a single other task, we train them two at a time, as shown in Table 8.2.

2. **Order of tasks**: To check whether the order of tasks affects the overall performance, we run sets of 3 and 4 tasks in various orders. We report the cases where the order has a measurable effect (positive or negative) on the performance.

3. **Context of words**: we are interested in measuring the effect of the context for SPELL (sequence-to-sequence). To do so, we either feed the data word by word or whole user-generated text at a time. In the following, SPELL will refer to the context-aware task, and SPELL-token refers to the contextless task.

4. **Size of SPELL training data**: we want to investigate the impact of the size of the training data, especially considering that one of the tasks (SPELL) has much more data than the other (CSD, SA and NER) tasks. To do so, we vary only the size of the training data of SPELL while keeping the training sets of CSD, SA and NER fixed each time (as well as the test sets).

5. **Training data augmentation**: We experiment with augmenting the training data for the SPELL task (further referred to as augmented). In this experiment, the training data is a combination of tokens and sequence of tokens. (This is equivalent to jointly training SPELL and SPELL-token.)

For each case we take models trained separately (single tasks) as baselines. For **pairwise tasks** we report the detailed results measured as the average Accuracy and macro F-score on the test sets over 50 epochs, thus taking into account the speed of learning. For other experiments (2, 3, 4, and 5) we show the performance, measured as the overall Accuracy, of jointly learning the tasks at hand on the test sets over 20 epochs (we found no significant gain when training for longer and do not report further).

5.1 Pairwise tasks

In Table 8.2, results measured as Accuracy indicate that learning SPELL, SA and NER tasks jointly with CSD improves their performance over learning them separately —by comparing the performance of single tasks to their performance when jointly trained with CSD.

Task	Tasks	Training	Accuracy (%)	Macro F-score
CSD	CSD	single	96.80	64.54
	CSD + SPELL	joint	96.32	62.27
	CSD + SA	joint	94.30	34.61
	CSD + NER	joint	**97.20**	**71.29**
SPELL	SPELL	single	93.49	
	SPELL + CSD	joint	**93.60**	
	SPELL + SA	joint	93.20	
	SPELL + NER	joint	**93.71**	
SA	SA	single	61.23	54.08
	SA + CSD	joint	**61.35**	53.31
	SA + SPELL	joint	60.74	51.50
	SA + NER	joint	59.82	53.46
NER	NER	single	99.80	49.68
	NER + CSD	joint	**99.82**	48.65
	NER + SPELL	joint	99.78	42.05
	NER + SA	joint	99.74	34.60

Table 8.2 Macro-average performance of the tasks trained separately and pairwise. Underlined values are baselines. Values in bold show positive effect of jointly learning the tasks at hand.

Note that the gain is mutual between CSD and NER, i.e., jointly learning the tasks benefits both, to different extents. Nevertheless, SPELL and SA slightly benefit from CSD but do not improve it. Interestingly whenever multi-task includes SPELL or SA tasks, the overall performance of the second task (CSD or NER) drops compared to learning the task separately. A closer look at the results per epoch in Figure 8.2 indicates that when beneficial, multi-task learning speeds up the performance of the tasks for the first few epochs.

The same behaviour is observed in experiments below (Section 5.2 for instance). This could be because (1) the generated shared representation is not wide enough to capture all tasks perfectly —it needs more parameters in shared layers, or that (2) each task has enough data in itself to reach maximum accuracy. (3) Another hypothesis, which contradicts (2), is that the sparsity and noise in the SPELL and SA training data effects negatively the other tasks.

Jointly training NER with CSD (in turquoise) outperforms training the tasks separately. Furthermore, jointly learning SA with CSD (in red) and SA with SPELL (in pink) outperforms SA trained as a single task. These observations refute hypothesis (2). We verified hypothesis (1) by increasing the number of features in the shared layers. That is to say, we tried different values and found that using 500 features in the CNN layers of the token-level encoder, and 500 and 1,000 features for the first and the second CNN layers of the sequence-level encoder has slightly improved the performance of SPELL. However, the overall behaviour of jointly learning SPELL or SA with CSD or NER is still the same.

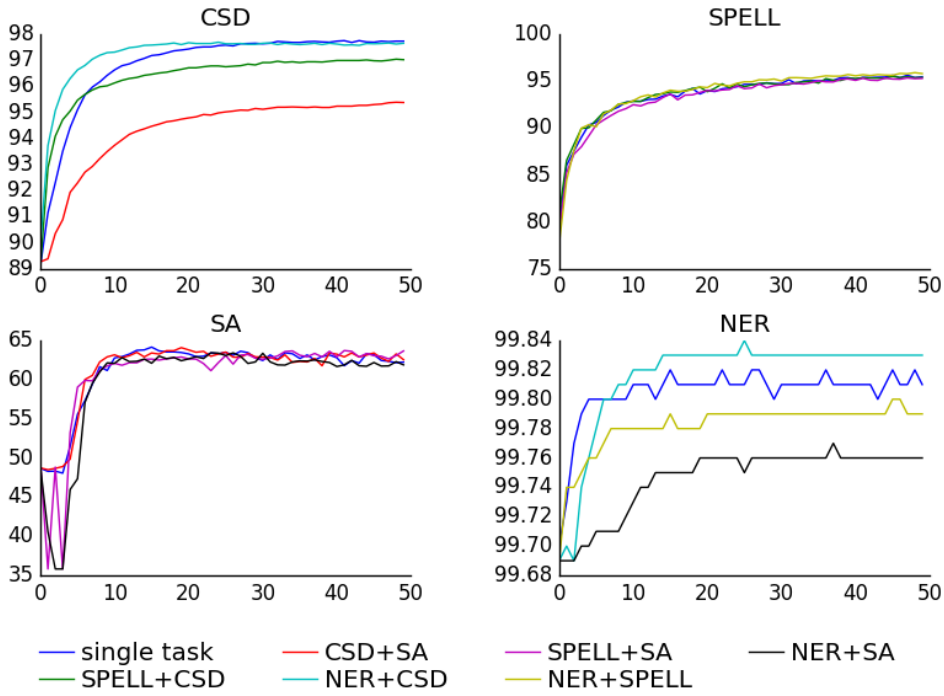


Figure 8.2 Accuracy (%) of jointly learning 2 tasks for 50 epochs.

This means that hypothesis (1) does not hold, i.e., it is likely that the noise and sparsity of the SPELL and SA datasets have negative effects on training them jointly with each other or with CSD and NER. This hypothesis requires further investigation, which we leave it as future work.

We provide in Table 8.2 the macro-average F-score for each setting which also reflects the overall impact of jointly learning the tasks by treating all the classes equally. Moreover, since all our datasets are imbalanced both in terms of class distributions and dataset sizes (certain classes have more samples than others and some datasets are much larger than others) we also show the micro F-score at a convergence point for each setting to better analyse the effect of jointly learning the tasks on each class.

Results in Table 8.3 show that jointly training CSD with SPELL or SA has negative effect on all CSD classes (marked with ↓). The negative effect of SA is more pronounced. Minority classes (BER, BOR, ENG, FRC, and NER) are more affected than others. Training CSD with NER has also caused some loss in the performance of some classes of CSD (marked with ↓), but the loss is smaller than when trained with SPELL or SA. The posi-

Task	Class	CSD	CSD + SPELL	CSD + SA	CSD + NER
CSD	ALG	92.05	89.82↓	83.90↓	91.86↓
	BER	74.29	71.43↓	00.00↓	64.71↓
	BOR	77.10	62.45↓	20.91↓	72.22↓
	DIG	99.93	99.93	99.25↓	99.93
	ENG	26.67	15.38↓	00.00↓	37.50↑
	FRC	83.62	74.45↓	44.93↓	82.17↓
	MSA	90.76	87.88↓	81.71↓	90.39↓
	NER	58.62	26.74↓	02.35↓	62.83↑
	SND	96.14	95.98↓	80.37↓	95.58↓
	Class	SA	SA + CSD	SA + SPELL	SA + NER
SA	MIX	60.38	62.48↑	64.20↑	60.70↑
	NEG	41.72	42.44↑	31.88↓	48.21↑
	NEU	53.80	50.95↓	54.95↑	56.11↑
	POS	75.59	75.92↑	76.92↑	75.48↓
	Class	NER	NER + CSD	NER + SPELL	NER + SA
NER	COM	21.54	29.55↑	11.32↓	00.00↓
	LOC	80.77	81.50↑	74.03↓	66.05↓
	OOO	99.50	99.59↑	99.45↓	99.42↓
	ORG	09.57	06.67↓	03.87↓	00.00↓
	OTH	26.39	27.41↑	22.66↓	18.75↓
	PER	63.38	69.77↑	54.36↓	52.17↓
	PRO	57.20	59.93↑	54.51↓	47.80↓

Table 8.3 Micro F-score of the tasks in single and multi-task settings. ↑ marks positive effect and ↓ marks negative effect of jointly learning the 2 tasks at hand.

tive effect of NER task on CSD (marked with ↑) could be attributed to its improvements for ENG and NER classes (two minor classes) with a gain of 10.83 and 4.21 points on the F-score respectively. One possible explanation for this improvement could be that the model could extract some underlying structures between some named entity mentions and English words used in the same context. It could be also that it becomes easier for the model to further classify a token in one of NER classes when it knows it is a named entity mention.

As shown in Table 8.3, some classes are harder to learn than others, even single trained models struggle with them. Overall SA benefits from CSD and NER. On the one hand, the gain from CSD could be attributed to its positive effect on MIX, NEG and POS classes. Nevertheless, CSD has negative effect on NEU with a loss of 2.85 points on the F-score. On the other hand, NER has improved MIX, NEG and NEU classes with a slight loss on POS. SPELL has improved MIX, NEU and POS and caused significant drop on NEG with a loss of 9.84 points on the F-score.

The main difference between the effect of the tasks is mainly on the minority classes (NEG and NEU). This suggests that the tasks could be complementary and their effect could be optimised if trained jointly. This is confirmed when trained the 4 tasks together

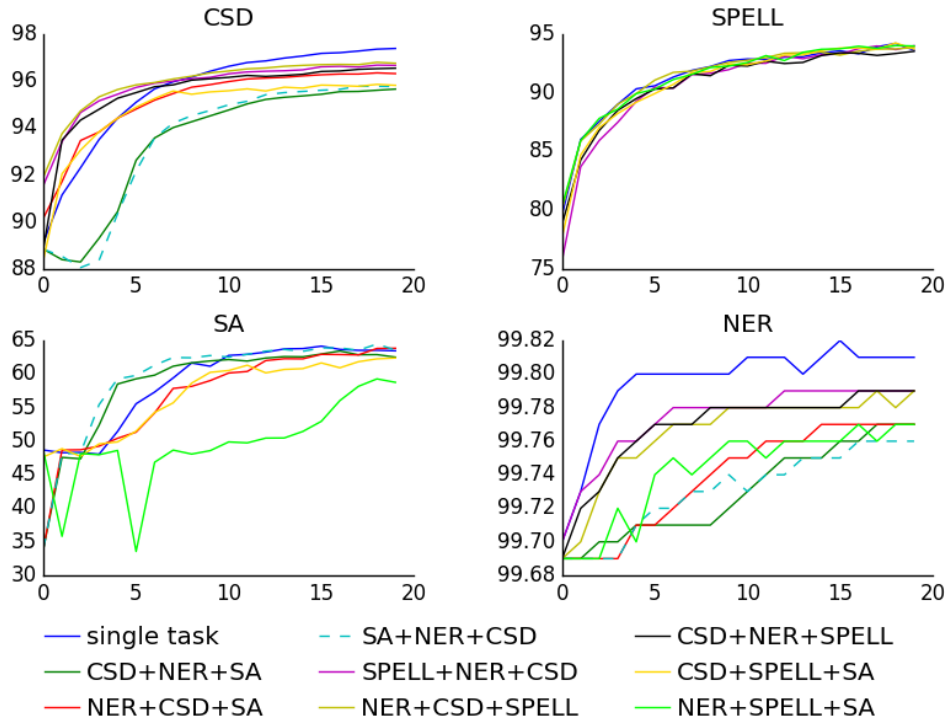


Figure 8.3 Accuracy (%) of jointly learning 3 tasks for 20 epochs with varying task order.

as shown in Figure 8.4 —at least for the first 10 epochs for SA.

SPELL and especially SA have significant negative impact on all classes of NER. Nonetheless CSD has improved all NER classes except ORG (which a single NER model already struggles to capture, with an F-score of only 9.57).

5.2 Order of tasks

Results in Figure 8.3 show that, except for NER, jointly learning the CSD, SPELL and SA tasks improves their performance over learning each one separately (as single tasks) only for the first few epochs (7 epochs), after that learning CSD as a single task outperforms training it with other tasks (blue line), and the effect of learning jointly the tasks is not clear for SPELL and SA.

The results suggest that the order of the tasks has a different effect on the different tasks, for the first few epochs. For instance, while training SA+NER+CSD has negative effect on both CSD and NER, it has positive effect on SA (outperforms even SA trained separately). Likewise for CSD-NER-SA but at different extent. NER+CSD+SA has negative effect on SA and NER overall, but it has positive effect on CSD at the beginning.

This suggests that the order of the tasks affects strongly the first epoch.

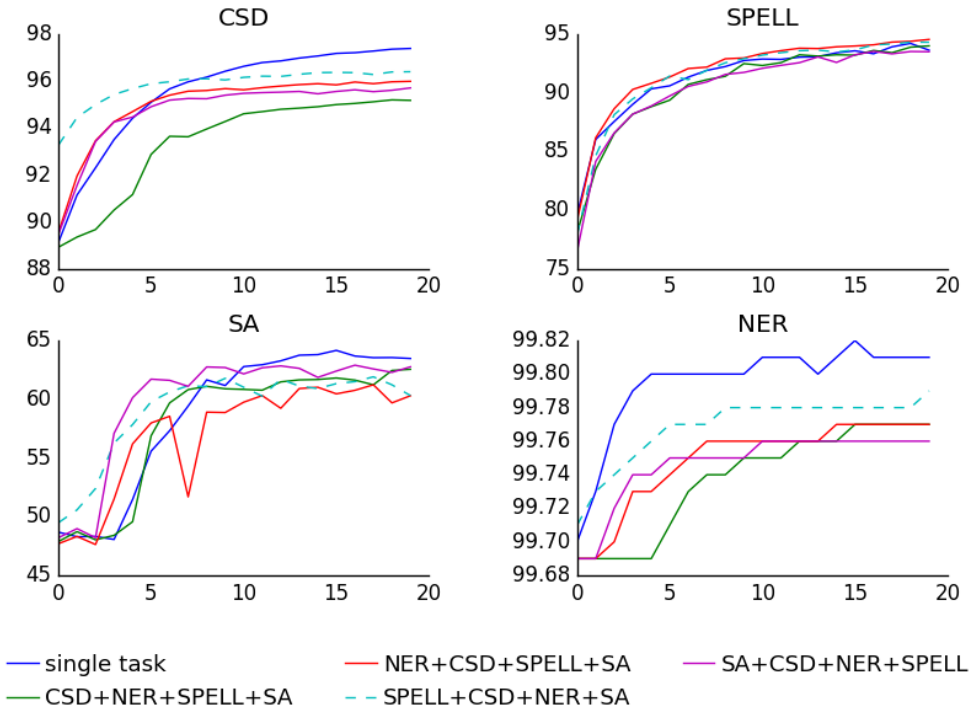


Figure 8.4 Accuracy (%) of jointly learning 4 tasks for 20 epochs with varying task order.

The same observation could be applied when jointly learning the 4 tasks as shown in Figure 8.4. In more details, jointly learning the 4 tasks in NER+CSD+SPELL+SA and SPELL+CSD+NER+SA orders improves SPELL where the task achieves its best performance. While the same task orders have no positive effect on NER, they do boost the performance of CSD and SA in the beginning and then they cause the overall performance to drop.

5.3 Context of words for SPELL

So far SPELL is trained at a sequence level (as a sequence-to-sequence). In order to measure the effect of the word context we train the same model architecture at a token level, and we refer to it as SPELL-token in Figure 8.5. The choice of NER+CSD+SPELL+SA order is based on the aforementioned results in Figure 8.4 where the selected task order performs the best for SPELL (in red). The results indicate clearly that context does matter for SPELL when trained separately and for CSD and NER tasks when trained jointly with SPELL and SA. Surprisingly, SPELL (with context) has a positive effect on SA only for

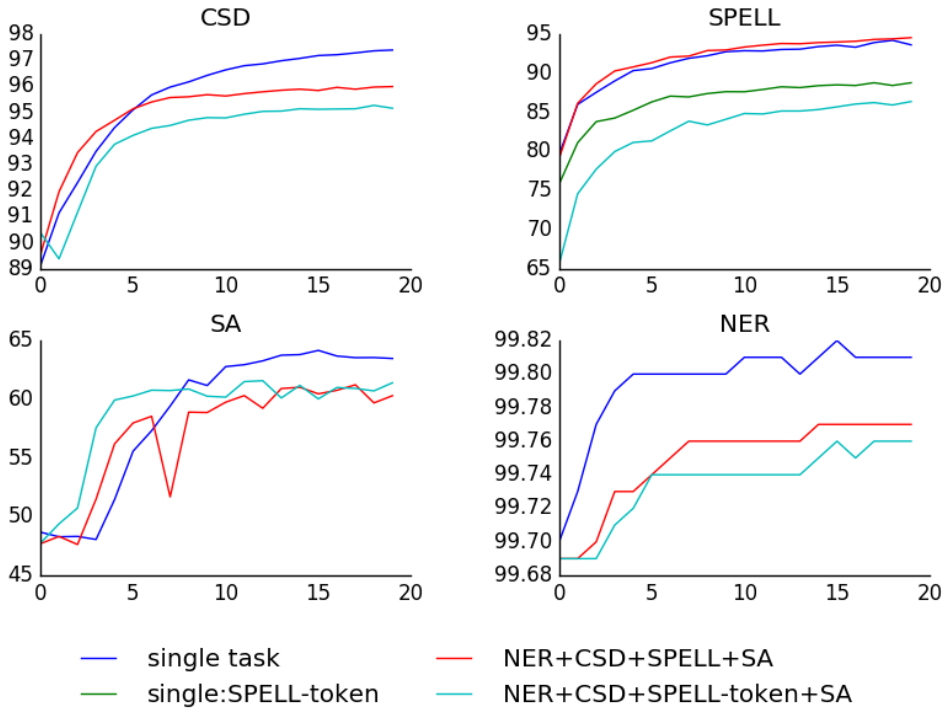


Figure 8.5 Accuracy (%) of jointly learning 4 tasks with(out) word context for SPELL.

the first 6 epochs then the effect is reversed. SPELL-token has an even more positive effect on SA before epoch 8. This suggests that either SA and SPELL datasets could include more ambiguity compared to other datasets, or that the noise of the two datasets hinders learning the tasks jointly.

5.4 Size of SPELL training data

As mentioned earlier, in this experiment we only vary the size of the training set for SPELL. We try 10k, 50k, 100k and all (>185k)) and keep the rest unchanged to investigate whether this has any impact on jointly learning the tasks. We use the same task order, namely NER+CSD+SPELL+SA as motivated earlier, and we refer to it as multi-task in Figure 8.6.

In single task learning, the learning curves of SPELL in Figure 8.6 indicate that the performance of the task improves quickly with more data (by comparing the performances of 100k to 10k and 50k training samples). However, the performance levels with 100k samples, even though it takes a few more epochs to reach the performance of when using all training data. Towards the end the two lines are almost superposed. One possible

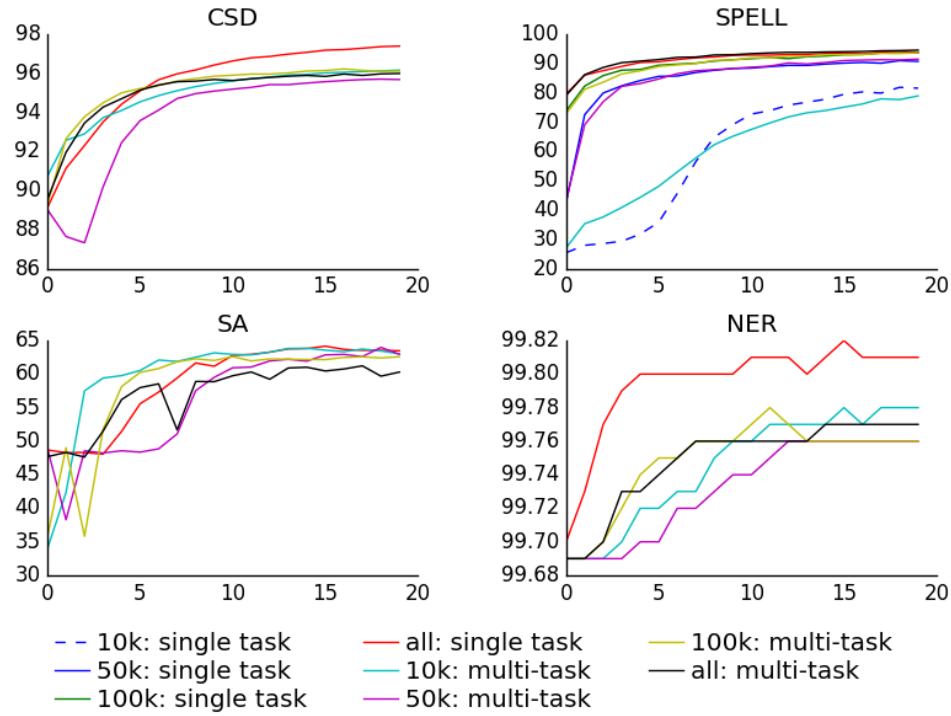


Figure 8.6 Accuracy (%) of jointly learning 4 tasks with varying SPELL training size. Single task: learning each task separately (baselines). Multi-task: jointly learning the tasks in NER+CSD+SPELL+SA order. All: train on all training sets as described in Section 5.

explanation is that most representative data is already covered in 100k (the model has already seen enough data to achieve its maximum performance).

In multi-task learning, the same trend of single task learning is observed for SPELL with a small gain in the performance in the beginning when multi-tasking. Interestingly, as the amount of data increases, the gain of multi-tasking diminishes. For CSD, increasing the training size of SPELL from 10k to 50k has a negative effect, but increasing the size to 100k has boosted the performance of CSD especially in the beginning. The same thing is observed for NER and SA. One possible explanation could be that the datasets of 50k or less are too small and subject to random noise.

The best gain of multi-task for CSD is achieved when trained with only 100k of SPELL. NER and SA, exceeding even single task, benefits the most when trained with only 10k of SPELL. SPELL nevertheless follows the "more data better performance" hypothesis.

5.5 Data augmentation

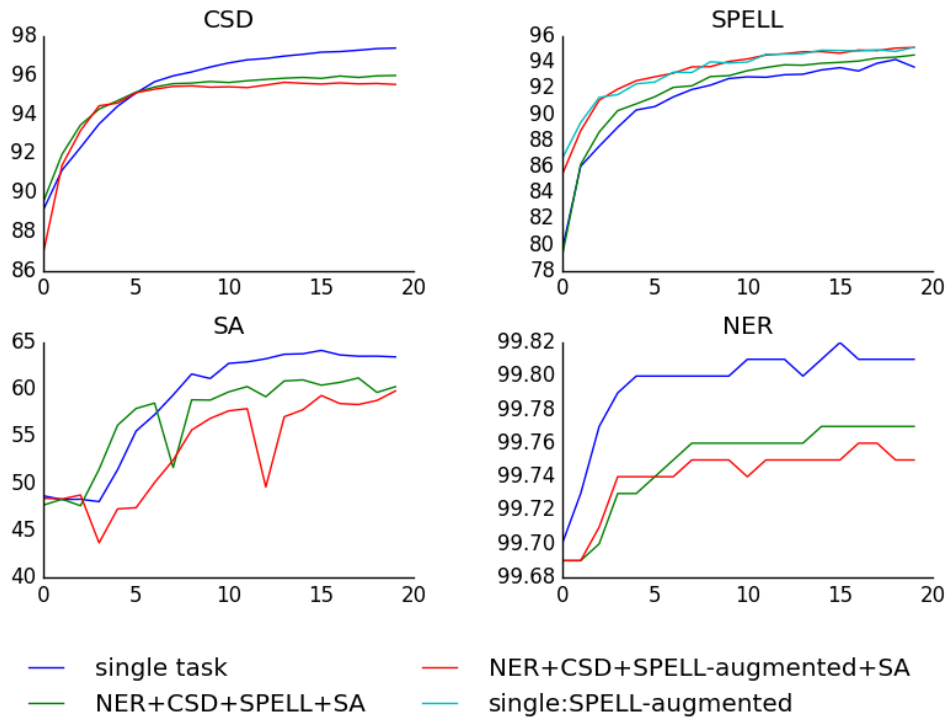


Figure 8.7 Accuracy (%) of jointly learning 4 tasks with data augmentation for SPELL. Augmented: using token + sequence as input to SPELL.

We replicate the same experiment as in Section 5.3, but instead of comparing the performance ofSPELL-token and SPELL separately, we augment the SPELL training data by combining both (token and sequence as input). This allows us to optimise the gain, if any, from the SPELL data.

Results in Figure 8.7 show that multi-task with the augmented data has arguably very little effect on SPELL compared to the single task in the same setting (the two lines are nearly superposed). However, data augmentation boosts the performance of SPELL compared to non-augmented data and even achieves its best performance. This rejects again hypothesis (2) in Section 5.1 because the performance of SPELL keeps increasing with more data.

On the one hand, augmenting SPELL data has a notable positive effect on SPELL when jointly trained with the other tasks compared to the same setting with non-augmented data (comparing green and red lines). On the other hand, in terms of effect on the other tasks, while augmenting SPELL data has a negative impact on SA, it offers a small benefit

for CSD and NER at the very beginning (before epoch 6), but it is outperformed by the non-augmented data after that.

6 Conclusions and Future Work

We have examined the effect of jointly learning 4 tasks, which are neither morpho-syntactic nor semantic tagging, for noisy user-generated Algerian texts. We described the tasks, namely CSD, SPELL, SA and NER, along with their corresponding datasets and model architectures. The main findings of our empirical investigation, which include a variety of experiments, could be summarised in the following points. (1) Tasks have different impacts on each other when learned jointly. (2) In multi-task learning notable gains are achieved for some tasks when trained jointly with specific tasks. Other tasks benefit from jointly learning them with some other tasks but the gain is only during the first few epochs, especially for tasks with little training data (CSD, NER and SA comparably to SPELL). Training for more epochs degraded their performance compared to learning them separately which is likely caused by the noisiness and sparsity of SPELL and SA data.

This means that it is hard to say whether multi-tasking is useful or not without mentioning several factors such as the tasks themselves, their order, the size of their datasets. (3) Word context for SPELL does matter for the task itself (single task) and for the tasks it is jointly trained with. (4) More SPELL training data does not necessary yield better results neither for the task itself (single task) nor for the tasks it is jointly learned with. In fact, performance is levelling at a certain point, in our case 10k for SA and NER, 100k for CSD, confirming this hypothesis. (5) Combining token and sequence level SPELL (augmented) is more beneficial for the task itself (single task) with no gain for multi-task at the convergence point.

In the future, we will examine hypothesis (3) using sequential transfer learning, for instance by running SPELL on all datasets and compare their performances to the non spell corrected and normalised ones. Furthermore, we plan to explore the idea of curriculum learning (Elman, 1993; Hacohen and Weinshall, 2019) on tasks and on individual classes for each task by introducing the tasks or the classes in increasing order of difficulty.

Chapter 9

Conclusions

1 Summary of the Thesis

The goal of this thesis was to explore the possibility of applying state-of-the-art approaches in NLP, namely deep neural networks (DNNs), to process languages in low-resource scenarios. More specifically to what extent DNNs trained end-to-end could be used to mitigate the lack of linguistic resources —labelled and unlabelled automated ready-to-use linguistic resources— and non-existence of processing tools in case linguistic resources are available. We took the case of the colloquial language used in Algeria —user-generated texts in social media— because it is low-resourced, and it presents interesting linguistic challenges, in particular, (i) high orthographic variability —non-standardised— and (ii) spontaneous code-switching.

In Chapter 1 we situated our work by (1) outlining our motivation, (2) explaining the research questions we attempt to address, (3) listing our contributions to the research field, and (4) highlighting the ethical considerations and the measures we have taken to minimise exposing, directly or indirectly, human subjects in our research to potential risks.

In Chapter 2 we dealt with the task of Code-switch Detection (CSD). As we see it language identification is the first step in any NLP pipeline, and our data is code-switched. We performed CSD at a token level because in our data source it is hard to define where a sentence starts or ends. Therefore we decided to do token level CSD then chunking if needed. We created benchmark linguistic resources and experimented with various taggers and setups. In Chapter 3 and Chapter 4 we explored other methods to improve CSD performance, including leveraging limited datasets, background knowledge, pretrained language models and/or embeddings.

In Chapter 5 we dealt with detecting Semantic Textual Similarity. In Chapter 6 we worked on normalising our data as a way to reduce its high orthographic variability. In Chapter 7 we attempted to identify sentiments from Algerian user-generated data with the same challenges. Finally, in Chapter 8 we explored the possibility of jointly learning the

earlier mentioned tasks. For each task we reported our experimental results which show that DNNs trained end-to-end is promising for languages in low-resource scenarios.

2 Future Directions

In the future, we would like to revisit multitask learning combined with curriculum learning and to include more tasks such as topic modeling and textual semantic similarity. We are particularly interested in topic modeling from a socio-linguistic perspective — knowing a topic of a discussion could help predict the (language) varieties in use (Sayahi, 2014). We expect that this setup might help reduce the variability in spelling words in non-standardised languages. Another future direction worth trying is to apply our language-independent models to other closely related varieties or other languages with the same settings. Moreover, since the majority of the work behind this thesis was done, the dominant way of building language models has shifted away from CNN and RNN, and moved to Transformer-based models (Vaswani et al., 2017) such as BERT (Devlin et al., 2019) and its variants. Nevertheless, these models are usually not trained with sparse low-resource data. Addressing this gap is a challenge to be addressed by the next generation of researchers.

Bibliography

Muhammad Abdul-Mageed and Mona Diab. 2012. AWATIF: A Multi-Genre Corpus for Modern Standard Arabic Subjectivity and Sentiment Analysis. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3907–3914, Istanbul, Turkey. European Language Resources Association (ELRA).

Wafia Adouane, Jean-Philippe Bernardy, and Simon Dobnik. 2018a. Improving Neural Network Performance by Injecting Background Knowledge: Detecting Code-switching and Borrowing in Algerian texts. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 20–28, Melbourne, Australia. Association for Computational Linguistics (ACL).

Wafia Adouane, Jean-Philippe Bernardy, and Simon Dobnik. 2019. Neural Models for Detecting Binary Semantic Textual Similarity for Algerian and MSA. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 78–87, Florence, Italy. Association for Computational Linguistics (ACL).

Wafia Adouane and Simon Dobnik. 2017. Identification of Languages in Algerian Arabic Multilingual Documents. In *Proceedings of the Third Arabic Natural Language Processing Workshop*, pages 1–8, Valencia, Spain. Association for Computational Linguistics (ACL).

Wafia Adouane, Simon Dobnik, Jean-Philippe Bernardy, and Nasredine Semmar. 2018b. A Comparison of Character Neural Language Model and Bootstrapping for Language Identification in Multilingual Noisy Texts. In *Proceedings of the Second Workshop on Subword/Character LEvel Models*, pages 22–31, New Orleans. Association for Computational Linguistics (ACL).

Wafia Adouane, Nasredine Semmar, Richard Johansson, and Victoria Bobicev. 2016. Automatic Detection of Arabicized Berber and Arabic Varieties. In *Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3)*, pages 63–72, Osaka, Japan. The COLING 2016 Organizing Committee.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iñigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 Task 2: Semantic

Textual Similarity, English, Spanish and Pilot on Interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 252–263, Denver, Colorado. Association for Computational Linguistics (ACL).

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. SemEval-2016 Task 1: Semantic Textual Similarity, Monolingual and Cross-Lingual Evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511, San Diego, California. Association for Computational Linguistics (ACL).

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393, Montréal, Canada. Association for Computational Linguistics (ACL).

Mohamed Aly and Amir Atiya. 2013. LABR: A Large Scale Arabic Book Reviews Dataset. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 494–498, Sofia, Bulgaria. Association for Computational Linguistics (ACL).

Ronald E. Anderson. 1992. ACM Code of Ethics and Professional Conduct. *Commun. ACM*, 35(5):94–99.

Rie Kubota Ando and Tong Zhang. 2005. A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data. *The Journal of Machine Learning Research*, 6:1817–1853.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint*, arXiv:1409.0473.

Utsab Barman, Amitava Das, Joachim Wagner, and Jennifer Foster. 2014a. Code Mixing: A Challenge for Language Identification in the Language of Social Media. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 13–23, Doha, Qatar. Association for Computational Linguistics (ACL).

Utsab Barman, Joachim Wagner, Grzegorz Chrupała, and Jennifer Foster. 2014b. DCU-UVT: Word-Level Language Classification with Code-Mixed Data. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 127–132, Doha, Qatar. Association for Computational Linguistics (ACL).

Jeremy Barnes, Samia Touileb, Lilja Øvrelid, and Erik Velldal. 2019. Lexicon Information in Neural Sentiment Analysis: a Multi-task Learning Approach. In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, pages 175–186, Turku, Finland. Linköping University Electronic Press.

Petr Baudis and Jan Šedivý. 2016. Sentence Pair Scoring: Towards Unified Framework for Text Comprehension. *arXiv preprint*, arXiv:1603.06127v4.

Islam Beltagy, Cuong Chau, Gemma Boleda, Dan Garrette, Katrin Erk, and Raymond Mooney. 2013. Montague Meets Markov: Deep Semantics with Probabilistic Logical Form. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 11–21, Atlanta, Georgia, USA. Association for Computational Linguistics (ACL).

Islam Beltagy, Katrin Erk, and Raymond Mooney. 2014. Probabilistic Soft Logic for Semantic Textual Similarity. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1210–1219, Baltimore, Maryland. Association for Computational Linguistics (ACL).

Emily M. Bender and Batya Friedman. 2018. Data Statements for Natural Language Processing: Toward Mitigating System Bias and Enabling Better Science. *Transactions of the Association for Computational Linguistics*, 6:587–604.

Joachim Bingel and Anders Søgaard. 2017. Identifying Beneficial Task Relations for Multi-task Learning in Deep Neural Networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 164–169, Valencia, Spain. Association for Computational Linguistics (ACL).

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *arXiv preprint*, arXiv:1607.04606v2.

Marcel Bollmann. 2019. A Large-scale Comparison of Historical Text Normalization Systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3885–3898, Minneapolis, Minnesota. Association for Computational Linguistics (ACL).

Marcel Bollmann, Anders Søgaard, and Joachim Bingel. 2018. Multi-Task Learning for Historical Text Normalization: Size Matters. In *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*, pages 19–24, Melbourne, Australia. Association for Computational Linguistics (ACL).

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A Large Annotated Corpus for Learning Natural Language Inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642. Association for Computational Linguistics (ACL).

Jean Carletta. 1996. Assessing Agreement on Classification Tasks: The Kappa Statistic. *Computational Linguistics*, 22(2):249–254.

Marine Carpuat. 2014. Mixed Language and Code-Switching in the Canadian Hansard. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 107–115, Doha, Qatar. Association for Computational Linguistics (ACL).

Rich Caruana. 1997. Multitask Learning. *Machine Learning*, 28:41–75.

Soravit Changpinyo, Hexiang Hu, and Fei Sha. 2018. Multi-Task Learning for Sequence Tagging: An Empirical Study. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2965–2977, Santa Fe, New Mexico, USA. Association for Computational Linguistics (ACL).

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for Natural Language Inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics (ACL).

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014a. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics (ACL).

Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv preprint*, arXiv:1406.1078.

François Chollet. 2015. Keras.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv preprint*, arXiv:1412.3555.

Ronan Collobert and Jason Weston. 2008. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA. ACM.

Council of European Union. 2016. REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 27 April 2016.

Abdelghani Dahou, Shengwu Xiong, Junwei Zhou, Mohamed Houcine Haddoud, and Pengfei Duan. 2016. Word Embeddings and Convolutional Neural Network for Arabic Sentiment Classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2418–2427, Osaka, Japan. The COLING 2016 Organizing Committee.

Amitava Das and Björn Gambäck. 2013. Code-Mixing in Social Media Text: The Last Language Identification Frontier? *Traitement Automatique des Langues*, 54(3):41–64.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics (ACL).

Kuntal Dey, Ritvik Shrivastava, and Saroj Kaushik. 2016. A Paraphrase and Semantic Similarity Detection System for User Generated Short-Text Content on Microblogs. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2880–2890. The COLING 2016 Organizing Committee.

Mona Diab, Pascale Fung, Mahmoud Ghoneim, Julia Hirschberg, and Thamar Solorio. 2016. Proceedings of the second workshop on computational approaches to code switching. Austin, Texas. Association for Computational Linguistics (ACL).

David Dittrich and Erin Kenneally. 2012. The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research.

William B. Dolan and Chris Brockett. 2005. Automatically Constructing a Corpus of Sentential Paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, South Korea.

Gabriel Doyle. 2014. Mapping Dialectal Variation by Querying Social Media. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 98–106. Association for Computational Linguistics (ACL).

Jacob Eisenstein. 2013. Phonological Factors in Social Media Writing. In *Proceedings of the Workshop on Language Analysis in Social Media*, pages 11–19, Atlanta.

Heba Elfardy, Mohamed Al-Badrashiny, and Mona Diab. 2013. Code Switch Point Detection in Arabic. In *Natural Language Processing and Information Systems*, pages 412–416, Berlin, Heidelberg. Springer Berlin Heidelberg.

Heba Elfardy and Mona Diab. 2012. Token Level Identification of Linguistic Code Switching. In *Proceedings of COLING 2012: Posters*, pages 287–296, Mumbai, India. The COLING 2012 Organizing Committee.

Jeffrey Elman. 1993. Learning and Development in Neural Networks: the Importance of Starting Small. *Cognition*, 48:71–99.

Jeffrey L. Elman. 1990. Finding Structure in Time. *Cognitive science*, 14(2):179–211.

Abdeljalil Elouardighi, Mohcine Maghfour, and Hafdalla Hammia. 2017. Collecting and Processing Arabic Facebook Comments for Sentiment Analysis. In *Model and Data Engineering*, pages 262–274, Cham. Springer International Publishing.

Hady ElSahar and Samhaa R. El-Beltagy. 2015. Building Large Arabic Multi-domain Resources for Sentiment Analysis. In *Computational Linguistics and Intelligent Text Processing*, pages 23–34, Cham. Springer International Publishing.

Ramy Eskander, Nizar Habash, Owen Rambow, and Nadi Tomeh. 2013. Processing Spontaneous Orthography. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 585–595. Association for Computational Linguistics (ACL).

Pravallika Etoori, Manoj Chinnakotla, and Radhika Mamidi. 2018. Automatic Spelling Correction for Resource-scarce Languages Using Deep Learning. In *Proceedings of Association for Computational Linguistics 2018, Student Research Workshop*, pages 146–152. Association for Computational Linguistics (ACL).

Charles Albert Ferguson. 1959. Diglossia. *WORD, Routledge*, 15(2):325–340.

Joshua Aaron Fishman. 1967. Bilingualism with and without Diglossia; Diglossia With and Without Bilingualism. *Journal of Social Issues*, 23:29 – 38.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional Sequence to Sequence Learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252, International Convention Centre, Sydney, Australia. PMLR.

Shaona Ghosh and Per Ola Kristensson. 2017. Neural Networks for Text Correction and Completion in Keyboard Decoding. *arXiv preprint*, arXiv: 1709.06429.

Imane Guellil, Ahsan Adeel, Faical Azouaou, and Amir Hussain. 2018. SentiALG: Automated Corpus Annotation for Algerian Sentiment Analysis. In *International Conference on Brain Inspired Cognitive Systems*, pages 557–567. Springer.

Nizar Habash, Fadhl Eryani, Salam Khalifa, Owen Rambow, Dana Abdulrahim, Alexander Erdmann, Reem Faraj, Wajdi Zaghouani, Houda Bouamor, Nasser Zalmout, Sara Hassan, Faisal Al shargi, Sakhar Alkhereyf, Basma Abdulkareem, Ramy Eskander, Mohammad Salameh, and Hind Saddiki. 2018. Unified Guidelines and Resources for Arabic Dialect Orthography. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Guy Hacohen and Daphna Weinshall. 2019. On The Power of Curriculum Learning in Training Deep Networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 2535–2544.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Dirk Hovy and Shannon L. Spruit. 2016. The Social Impact of Natural Language Processing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 591–598, Berlin, Germany. Association for Computational Linguistics (ACL).

Minqing Hu and Bing Liu. 2004. Mining and Summarizing Customer Reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Philip Jansen, Wessel Reijers, David Douglas, Faridun Sattarov, Agata Gurzawska, Alexandra Kapeller, Philip Brey, Rok Benčin, Zuzanna Warso, and Robert Braun. 2017. A Reasoned Proposal for Shared Approaches to Ethics Assessment in the European Context.

Mohamed Amine Jerbi, Hadhemi Achour, and Emna Souissi. 2019. Sentiment Analysis of Code-Switched Tunisian Dialect: Exploring RNN-Based Techniques. In *International Conference on Arabic Language Processing*, pages 122–131. Springer.

Anna Jørgensen, Dirk Hovy, and Anders Søgaard. 2015. Challenges of Studying and Processing Dialects in Social Media. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 9–18. Association for Computational Linguistics (ACL).

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of Tricks for Efficient Text Classification. *arXiv preprint*, arXiv:1607.01759.

Rudolf Kadlec, Martin Schmid, and Jan Kleindienst. 2015. Improved Deep Learning Baselines for Ubuntu Corpus Dialogs.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics (ACL).

Tom Kocmi and Ondřej Bojar. 2017. Lanidenn: Multilingual Language Identification on Text Stream. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 927–936. Association for Computational Linguistics (ACL).

Zeyang Lei, Yujiu Yang, Min Yang, and Yi Liu. 2018. A Multi-sentiment-resource Enhanced Attention Network for Sentiment Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 758–763, Melbourne, Australia. Association for Computational Linguistics (ACL).

Chen Liang, Praveen Paritosh, Vinodh Rajendran, and Kenneth D. Forbus. 2016. Learning Paraphrase Identification with Structural Alignment. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 2859–2865. AAAI Press.

Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Łukasz Kaiser. 2016. Multi-task Sequence to Sequence Learning. In *International Conference on Learning Representations (ICLR)*.

Roy Coleman Major. 2002. The Bilingualism Reader. Li Wei (ed.). London: Routledge, 2000. *Studies in Second Language Acquisition*, 24:491 – 493.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK Cure for the Evaluation of Compositional Distributional Semantic Models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA).

Héctor Martínez Alonso and Barbara Plank. 2017. When is Multitask Learning Effective? Semantic Sequence Prediction under Varying Data Conditions. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational*

Linguistics: Volume 1, Long Papers, pages 44–53, Valencia, Spain. Association for Computational Linguistics (ACL).

M´hamed Mataoui, Omar Zelmati, and Madiha Boumechache. 2016. Aproposed Lexicon-based Sentiment Analysis Approach for the Vernacular Algerian Arabic. *Research in Computing Science*, 110:55–70.

Salima Medhaffar, Fethi Bougares, Yannick Estève, and Lamia Hadrich-Belguith. 2017. Sentiment Analysis of Tunisian Dialects: Linguistic Ressources and Experiments. In *Proceedings of the Third Arabic Natural Language Processing Workshop*, pages 55–61, Valencia, Spain. Association for Computational Linguistics (ACL).

James Milroy. 2001. Language Ideologies and the Consequence of Standardization. *Journal of Sociolinguistics*, 5:530 – 555.

Behrang Mohit, Alla Rozovskaya, Nizar Habash, Wajdi Zaghouani, and Ossama Obeid. 2014. The First QALB Shared Task on Automatic Text Correction for Arabic. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pages 39–47, Doha, Qatar. Association for Computational Linguistics (ACL).

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese Recurrent Architectures for Learning Sentence Similarity. In *AAAI*, pages 2786–2792. AAAI Press.

Mahmoud Nabil, Mohamed Aly, and Amir Atiya. 2015. ASTD: Arabic Sentiment Tweets Dataset. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2515–2519, Lisbon, Portugal. Association for Computational Linguistics (ACL).

Dong Nguyen and A. Seza Doğruöz. 2013. Word Level Language Identification in Online Multilingual Communication. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 857–862, Seattle, Washington, USA. Association for Computational Linguistics (ACL).

Rodney d. Nielsen, Wayne Ward, and James h. Martin. 2009. Recognizing Entailment in Intelligent Tutoring Systems*. *Nat. Lang. Eng.*, 15(4):479–501.

Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. 2000. Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning, Special issue on information retrieval,*, 39:103–134.

Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345 – 1359.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A Decomposable Attention Model for Natural Language Inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255. Association for Computational Linguistics (ACL).

Nanyun Peng and Mark Dredze. 2017. Multi-task Domain Adaptation for Sequence Tagging. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 91–100, Vancouver, Canada. Association for Computational Linguistics (ACL).

Eva Pettersson. 2016. Spelling Normalisation and Linguistic Analysis of Historical Text for Information Extraction. *PhD thesis*, Studia Liguistica Upsaliensia 17, Uppsala: Acta Universitatis Upsaliensis.

David Pierce and Claire Cardie. 2001. Limitations of Co-Training for Natural Language Learning from Large Datasets. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mario Piergallini, Rouzbeh Shirvani, Gauri S. Gautam, and Mohamed Chouikha. 2016. Word-Level Language Identification and Predicting Codeswitching Points in Swahili-English Language Data. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 21–29, Austin, Texas. Association for Computational Linguistics (ACL).

Shana Poplack and Marjory Meechan. 1998. How Languages Fit Together in Codemixing. *The International Journal of Bilingualism*, 2(2):127–138.

Qiao Qian, Minlie Huang, Jinhao Lei, and Xiaoyan Zhu. 2017. Linguistically Regularized LSTM for Sentiment Classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1679–1689, Vancouver, Canada. Association for Computational Linguistics (ACL).

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.

John Rickford. 1990. Dialects in Contact. *Language in Society - Oxford and New York: Basil Blackwell, 1986*, 19.

Mohammed Rushdi-Saleh, M Teresa Martín-Valdivia, L Alfonso Ureña-López, and José M Perea-Ortega. 2011. OCA: Opinion Corpus for Arabic. *Journal of the American Society for Information Science and Technology*, 62(10):2045–2054.

Houda Saadane and Nizar Habash. 2015. A Conventional Orthography for Algerian Arabic. In *Proceedings of the Second Workshop on Arabic Natural Language Processing, ANLP, ACL 2015, Beijing, China, July 30, 2015*, pages 69–79. Association for Computational Linguistics (ACL).

Mohammad Salameh, Saif Mohammad, and Svetlana Kiritchenko. 2015. Sentiment after Translation: A Case-study on Arabic Social Media Posts. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 767–777, Denver, Colorado. Association for Computational Linguistics (ACL).

Younes Samih, Suraj Maharjan, Mohammed Attia, Laura Kallmeyer, and Thamar Solorio. 2016. Multilingual Code-switching Identification via LSTM Recurrent Neural Networks. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 50–59, Austin, Texas. Association for Computational Linguistics (ACL).

Younes Samih and Wolfgang Maier. 2016. Detecting Code-switching in Moroccan Arabic. In *Proceedings of SocialNLP @ IJCAI-2016*.

Lotfi Sayahi. 2014. *Diglossia and Language Contact: Language Variation and Change in North Africa*. Cambridge Approaches to Language Contact. Cambridge University Press.

Tyler Schnoebelen. 2013. *The Weirdest Languages*. Corpus Linguistics, retrieved on January 2020.

Burr Settles. 2009. Active Learning Literature Survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

Khaled Shaalan, Mohammed Attia, Pavel Pecina, Younes Samih, and Josef van Genabith. 2012. Arabic Word Generation and Modelling for Spell Checking. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Bonggun Shin, Timothy Lee, and Jinho D. Choi. 2017. Lexicon Integrated CNN Models with Attention for Sentiment Analysis. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 149–158, Copenhagen, Denmark. Association for Computational Linguistics (ACL).

Anil Kumar Singh and Jagadeesh Gorla. 2007. Identification of Languages and Encodings in a Multilingual Document. In *Building and Exploring Web Corpora (WAC3-2007): Proceedings of the 3rd Web as Corpus Workshop, Incorporating Cleaneval*.

Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steven Bethard, Mona Diab, Mahmoud Ghoneim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirschberg, Alison Chang, and Pascale Fung. 2014. Overview for the First Shared Task on Language Identification in Code-Switched Data. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 62–72, Doha, Qatar. Association for Computational Linguistics (ACL).

S. Sooraj, K. Manjusha, M. Kumar, and K.P. Soman. 2018. Deep Learning Based Spell Checker for Malayalam Language. *Journal of Intelligent and Fuzzy Systems*, 34:1427–1434.

Lameen Souag. 2000. *Grammar of Algerian Darja*. Retrieved on September 2016.

Assia Soumeur, Mheni Mokdadi, Ahmed Guessoum, and Amina Daoud. 2018. Sentiment Analysis of Users on Social Networks: Overcoming the Challenge of the Loose Usages of the Algerian Dialect. *Procedia computer science*, 142:26–37.

Bird Steven, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. 2007. Dynamic Conditional Random Fields: Factorized Probabilistic Models for Labeling and Segmenting Sequence Data. *Journal of Machine Learning Research*, (8(Mar)):693–723.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based Methods for Sentiment Analysis. *Computational linguistics*, 37(2):267–307.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China. Association for Computational Linguistics (ACL).

Jörg Tiedemann. 2009. Character-based PSMT for Closely Related Languages. In *Proceedings of 13th Annual Conference of the European Association for Machine Translation*, pages 12–19, Barcelona, Spain.

Peter D Turney. 2002. Thumbs up or Thumbs down?: Semantic Orientation Applied to Unsupervised Classification of Reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 417–424. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and Tell: A Neural Image Caption Generator. In *Computer Vision and Pattern Recognition*.

Ekaterina Vylomova, Trevor Cohn, Xuanli He, and Gholamreza Haffari. 2017. Word Representation Models for Morphologically Rich Languages in Neural Machine Translation. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 103–108. Association for Computational Linguistics (ACL).

Janet C. E. Watson. 2007. *The Phonology and Morphology of Arabic*. The Phonology of the World's Languages. OUP Oxford.

Wei Xu, Chris Callison-Burch, and Bill Dolan. 2015. SemEval-2015 Task 1: Paraphrase and Semantic Similarity in Twitter (PIT). In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 1–11, Denver, Colorado. Association for Computational Linguistics (ACL).

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A Challenge Dataset for Open-Domain Question Answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018. Association for Computational Linguistics (ACL).

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. *Transactions of the Association for Computational Linguistics*.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How Transferable are Features in Deep Neural Networks? In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3320–3328. Curran Associates, Inc.

Rabih Zbib, Erika Malchiodi, Jacob Devlin, David Stallard, Spyros Matsoukas, Richard Schwartz, John Makhoul, Omar F. Zaidan, and Chris Callison-Burch. 2012. Machine Translation of Arabic Dialects. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 49–59, Montréal, Canada. Association for Computational Linguistics (ACL).

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. *Character-level Convolutional Networks for Text Classification*. Advances in Neural Information Processing Systems 28 (NIPS 2015).

Dengyong Zhou, Olivier Bousquet, Thomas N. Lal, Jason Weston, and Bernhard Schölkopf. 2004. Learning with Local and Global Consistency. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 321–328. MIT Press.