NEURAL SEQUENCE LABELING ON SOCIAL MEDIA TEXT

by
Gustavo Aguilar

A proposal submitted to the Department of Computer Science,

College of Natural Sciences and Mathematics

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in Computer Science

Chair of Committee: Thamar Solorio

Committee Member: Ioannis Kakadiaris

Committee Member: Rakesh Verma

Committee Member: Mona Diab

University of Houston
December 2020

ACKNOWLEDGMENTS

This dissertation is the culmination of a unique and unforgettable experience in my life. Coming to the U.S. in 2016 after living my whole life in El Salvador was a challenging and exciting experience. The culture shock, the language barrier, and leaving my family and friends back in my home country definitely brought difficult times. However, I was lucky to meet wonderful people during my Ph.D., and I sincerely thank all of them for supporting me throughout my academic journey.

First and foremost, I want to thank my advisor, Dr. Thamar Solorio, for her endless support and advice throughout my graduate studies at UH. I will remember many advisor-advisee stories, but the day you allowed me to join your lab, you absolutely changed my career. Even though I had zero knowledge about research and natural language processing, you still gave me a real shot to grow and succeed in this field. I am deeply grateful for that and for your strong drive to support minorities. I admire your passion and enthusiasm for research, but I will never forget who you are as a person.

I also want to thank my dissertation committee members, Dr. Mona Diab (GWU and Facebook AI), Dr. Ioannis Kakadiaris (UH), and Dr. Rakesh Verma (UH), for their support, guidance, and feedback to make this dissertation better every time. I am also thankful to my colleagues at the RiTUAL lab for all the discussions and learning experiences. Particularly, I want to thank Suraj Maharjan and Sudipta Kar. I frequently engaged in insightful discussions and technical deep dives with them in our research projects.

Furthermore, I was lucky to have three fantastic summer research internships during my Ph.D. In 2018, I enjoyed working with Viktor Rozgić (mentor), Weiran Wang, and Chao Wang (manager) at Amazon Alexa Speech in Boston. In 2019, I had the opportunity to work with Yuan Ling (co-mentor), Yu Zhang (co-mentor), Benjamin Yao (manager), Xing Fan, and Edward Guo at Amazon Alexa AI in Seattle. In 2020, my last internship was remotely due to the COVID-19 pandemic, but that was not an impediment to have a great experience

with Bryan McCann (manager), Tong Niu, and Nitish Keskar. I also want to give a special mention to Leonardo Neves, who I got very fortunate to collaborate with after receiving the 2018 Snap Research Fellowship award. I thank you all for the countless learning experiences that have made me a better researcher every time.

Finally, I want to thank my family for their continuous support from start to end of this degree. To my parents, José Antonio Aguilar and María Ermilda Alas, despite being many miles away, I felt your support every time I needed it. To my siblings, Gerardo and Marcela Aguilar, you both inspire me with your life experiences, and I am grateful for all the support you provided me. I also want to thank my fiancée, Daniela De la Parra, for her unconditional support and limitless patience throughout my Ph.D. You have been my driving force and an encouraging partner during my academic journey's ups and downs.

ABSTRACT

As social media (SM) brings opportunities to study societies across the world, it also brings a variety of challenges to automate the processing of SM language. In particular, most of the textual content in SM is considered *noisy*; it does not always stick to the rules of the written language, and it tends to have misspellings, arbitrary abbreviations, orthographic inconsistencies, and flexible grammar. Additionally, SM platforms provide a unique space for multilingual content. This polyglot environment requires modern systems to adapt to a diverse range of languages, imposing another linguistic barrier to processing and understanding of text from SM domains.

This dissertation aims at providing novel sequence labeling approaches to handle noise and linguistic code-switching (i.e., the alternation of languages in the same utterance) in SM text. In particular, the first part of this dissertation focuses on named entity recognition for English SM text, where I propose linguistically-inspired methods to address phonological writing and flexible syntax. Besides, I investigate whether the performance of current state-of-the-art models relies on memorization or contextual generalization of entities.

In the second part of this dissertation, I focus on three sequence labeling tasks for code-switched SM text: language identification, part-of-speech tagging, and named entity recognition. Specifically, I propose transfer learning methods from state-of-the-art monolingual and multilingual models, such as ELMo and BERT, to the code-switching setting for sequence labeling. These methods reduce the demand for code-switching annotations and resources while exploiting multilingual knowledge from large pre-trained unsupervised models.

The methods presented in this dissertation are meant to benefit higher-level NLP applications oriented to social media domains, including but not limited to question-answering, conversational systems, and information extraction.

TABLE OF CONTENTS

ACKNOWLEDGMENTS . . . iii
ABSTRACT . . . v
LIST OF TABLES . . . xii
LIST OF FIGURES . . . xv

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Research Objectives . . . 4
1.3 Overview of Proposed Research . . . 5
1.3.1 Named Entity Recognition . . . 5
1.3.2 Linguistic Code-Switching . . . 6
1.4 Contributions . . . 8
1.5 Publications . . . 9

2 Literature Review **12**
2.1 Named Entity Recognition . . . 13
2.1.1 Hand-Crafted Features and Rule-based Systems . . . 13
2.1.2 Adapting NER Systems to Social Media Text . . . 14
2.1.3 Neural Networks for NER . . . 15
2.1.4 Pre-trained Language Models for NER . . . 16
2.2 Linguistic Code-Switching . . . 16
2.3 The Research Direction . . . 18

I Named Entity Recognition on Social Media Text **19**

3 Modeling Social Media Noise **20**
3.1 Methodology . . . 22
3.1.1 Feature Representation . . . 22
3.1.2 Model Architecture . . . 23
3.2 Experimental Setting . . . 27
3.2.1 Dataset . . . 27
3.2.2 Implementation Details . . . 28

3.3 Results and Analysis . . . 29
3.4 Limitations . . . 33
3.5 Conclusion . . . 34

4 Adapting to Flexible Syntax 35
4.1 Methodology . . . 36
4.1.1 Feature Representation . . . 36
4.1.2 Model Architecture . . . 37
4.2 Experiments and Results . . . 41
4.3 Analysis . . . 44
4.4 Limitations . . . 50
4.5 Conclusion . . . 50

5 Are Models Relying on Memorization or Generalization? 52
5.1 Data Analysis . . . 54
5.2 Model Analysis: Fine-tuned BERT . . . 56
5.3 Reducing the Memorization Bias . . . 60
5.3.1 Results . . . 62
5.4 Analysis . . . 64
5.5 Limitations . . . 68
5.6 Conclusions and Future Work . . . 68

II Sequence Labeling on Linguistic Code-Switching 69

6 LinCE: A Centralized Linguistic Code-Switching Evaluation Benchmark 70
6.1 Linguistic Challenges . . . 71
6.2 Tasks . . . 74
6.2.1 Language Identification (LID) . . . 74
6.2.2 Parts-of-Speech (POS) Tagging . . . 76
6.2.3 Named Entity Recognition (NER) . . . 78
6.2.4 Sentiment Analysis (SA) . . . 79
6.2.5 Stratification . . . 80
6.2.6 Evaluation . . . 83
6.3 Limitations . . . 84
6.4 Conclusion . . . 84

7 From English to Code-Switching 85
7.1 Methodology . . . 86
7.1.1 Position-aware Hierarchical Attention . . . 86
7.1.2 Sequence Tagging . . . 89
7.2 LID Experiments . . . 90
7.3 POS Tagging and NER Experiments . . . 92

7.4 Analysis . . . 94
7.5 Limitations . . . 96
7.6 Conclusion . . . 96

8 From Multilingualism to Code-Switching **97**
8.1 Method . . . 99
8.1.1 Approximating the Subword Embedding Table . . . 100
8.1.2 Pre-training with the Char2subword Module . . . 103
8.1.3 Fine-tuning . . . 105
8.2 Experiments . . . 107
8.2.1 Embedding Approximation . . . 107
8.2.2 Fine-tuning Experiments . . . 110
8.3 Analysis . . . 111
8.4 Limitations . . . 117
8.5 Conclusion . . . 118

9 Conclusions **119**
9.1 English-oriented Methods . . . 120
9.2 Code-switching-oriented Methods . . . 121
9.3 Detailed Contributions . . . 123
9.4 Future Work . . . 127
BIBLIOGRAPHY . . . 130

LIST OF TABLES

2.1 Results of different NER competitions. The performance degrades as the systems are moved to social media (SM) environments. The last row considers multiple SM domains, such as Twitter, YouTube, Reddit, and StackExchange. 15

3.1 Noisy and normalized text with equal International Phonetic Alphabet mappings. 23
3.2 WNUT 2017 dataset with the class frequency distribution. 27
3.3 The class-level and overall results of the systems on the WNUT 2017 dataset. **Stacked** means that the model was trained on a two-phase setting (i.e., first the MTL network, then the CRF classifier). **E2E** means that the model was trained in a single phase, back-propagating the gradients from the CRF down to the feature extractor. **WNUT** represents the winning system of the shared task by the UH-RiTUAL team in 2017. A t-test experiment shows that the stacked model improvement over the end-to-end model is statistically significant, with p-value < 0.0025 [38]. 29
3.4 I performed an ablation experiment on the stacked model. The results in the table are the average of the scores of three iterations. 31
3.5 Model predictions of the Reddit text in WNUT 2017 dataset. The bold words are the gold labels, and the underlined words are the predictions. 33

4.1 The data splits on each corpus along with the number of named entities. . . 42
4.2 The F_1 scores on the validation sets. BLSTM and TLSTM are **b**idirectional and **t**ree LSTMs. † denotes residual connections on every component. RA and GA refer to relative and global attentions. All the experiments use ELMo embeddings and CRF. Running t-test shows that the experiment numbers between methods (BLSTM vs. TLSTM) are statistical significant with all p-value < 0.01. The experiment 3.1 on the SemEval-2010 dataset is also statistically significant against the previous best score given by experiment 2.6. 43
4.3 The most attended words for nouns that are labeled as `PERSON` or `LOCATION`. The table shows the attended POS tag per entity, its coverage, and the corresponding list of words. 48

5.1 Statistics of the entity overlap with respect to the test set. **Overlap** describes the set operations: (Train ∪ Development) ∩ Test. **Repeated** means that the frequency of every entity is considered in the calculation, while **Unique** disregards the frequencies. Notably, the overlap percentages of the CoNLL-2003 are substantially higher than the ones from WNUT-2016. **Diversity** refers to the percentage of total unique entities out of the total repeated entities (e.g., 100% means all entities instances appear once). I sorted rows by overlap percentage from the repeated entities. 55
5.2 Results on the full test set and its observed and unobserved entity subsets. For the observed subsets, ≥ 1 means that the entities appear in training at least once, whereas ≤ 5 refers to entities that appear in training at most five times. Note that there is a decreasing tendency (↓) in the scores of the unobserved entities with respect to the full dataset scores; the scores from the observed entities tend to increase (↑). The "N/A" entries for movie and TV show mean that the classes do not appear in such subsets. 58
5.3 Results of the proposed methods to reduce memorization. **Obs.** and **Unobs.** refer to the observed and unobserved entity subsets, and **All** means the full test set. I show the average of three runs with different random seeds and their std. deviation as a subscript. 62
5.4 Class-level F_1 scores for the baseline and the frequency loss weight (FLW) experiments on the unobserved test set. I also provide the number of instances and the diversity percentage per class. The rows are sorted descendingly by the number of instances. Note that the diversity rate is the lowest for the company class, while the scores per model are almost the same. 65
5.5 The table shows cases where the FLW model gets the predictions wrong. The sentences are tagged with their corresponding ground-truth entities in brackets (e.g., $[\text{Penders Field}]_{\text{facility}}$). Also, I provide the results of the baseline for comparison. 66

6.1 Overview of the LinCE language pairs and tasks. 71
6.2 The CMI scores and the number of tokens across corpora. **All Posts** describes the number of posts in the corpora and **All CMI** is the corresponding CMI scores for such samples. Similarly, **CS Posts** denotes the number of code-switched posts (excluding monolingual posts) and **CS CMI** is the corresponding CMI scores for such samples. I also show the number of tokens that belong to the language pairs (**Lang1**, **Lang2**) as well as the overall number of tokens (**All Tokens**), which includes other LID labels beyond the language pairs. English is the **Lang1** class for English-paired languages; for MSA-EA, Modern Standard Arabic is the **Lang1** class. I omit the CMI information for the MSA-EA NER corpus because the corpus does not come with language identification labels. 72

6.3 Final data distribution of the LinCE benchmark. Note that the proposed distribution follows the stratification process described in Section 6.2.5, which generates partitions that differ from the original datasets. 81
6.4 The table shows the datasets for which I propose new splits. The column **Reason** provides the reason number according to the aspects listed in Section 6.2.5. The lower the KL-divergence, the more similar the splits are to the full corpus distribution. 82
6.5 The LinCE leaderboard as of October 19th, 2020. The top three entries in the table are the baseline models using off-the-shelf models publicly available. The char2subword mBERT model is a sequence labeling method that I proposed in Chapter 8. The missing scores denote that the participants did not provide results on those task. Hence, the average score is also skipped. 83

7.1 The results of incremental experiments on each LID dataset. The scores are calculated using the weighted F-1 metric across the eight LID labels from CALCS. Within each column, the best score in each block is in **bold**, and the best score for the whole column is underlined. Note that development scores from subsequent experiments (e.g., experiments 2.2 and 2.3) are statistically significant with p-value < 0.02. 91
7.2 The F_1 scores on POS tagging for the Hindi-English dataset. CS knowledge means that the CS-ELMo architecture has been adapted to code-switching by using the LID task. 93
7.3 The F_1 scores on the Spanish-English NER dataset. CS knowledge means that the CS-ELMo architecture has been adapted to code-switching by using the LID task. 94

8.1 Single-character operations to incorporate noise in the approximation stage. The operations are applied to every word in the vocabulary that exceeds the four characters and that it is not a special token. 104
8.2 The results of approximating the subword embedding table from mBERT using different combinations of objective functions. Experiments 1.1 to 1.4 denote the performance of the char2subword module using individual objective functions (e.g., experiment 1.2 only uses $\mathcal{L}_{cos}$). Experiments 1.5 to 1.8 use the cross-entropy objective $\mathcal{L}_{ce}$ by default and combine it with other objectives (e.g., experiment 1.8 uses $\mathcal{L}_{ce}(\cdot) + \mathcal{L}_{neigh}(\cdot)$). Experiment 1.9 combines all the objectives at the same time. The accuracy denotes the capability of the model to predict a subword out of its characters. Precision @ k measures the overlap between the k ground-truth neighbors for a vector $\boldsymbol{e}_i$ (that represents subword s_i) and the k neighbors of the predicted vector $\hat{\boldsymbol{e}}_i$. 108

8.3 The results of approximating the subword embedding table from BERT. The first row describes the neighbors of the query vectors coming from the same embedding table $\boldsymbol{E}$. Note that for words that do not exist in the table, I only show how the word would be tokenized. The other rows contain neighbors in every cell since they always have representations for the sequence of characters. 109
8.4 Results on the development set of the LinCE benchmark (average over three runs with different seeds). Full refers to the full mode where the model only uses the char2subword to embed the input. Hybrid means that the model uses the subword embedding table by default and backs off to the char2subword module for unseen words (i.e., out-of-vocabulary words) instead of splitting it. For this table, pre-trained means that the model was approximated after the pre-training phase (i.e., "Approx. → Pre-training → Approx."), while approx. means that the model is only trained in the approximation phase. The languages involved are English (en), Spanish (es), Hindi (hi), Nepali (ne), Modern Standard Arabic (msa), and Egyptian Arabic (arz). The scores for LID, NER, and POS are weighted F_1, micro F_1 with spans, and accuracy, respectively. The best results on each language pair are in bold. 111
8.5 Results on the test set of the LinCE benchmark using the hybrid char2subword mBERT model (best proposed model). The languages involved are English (en), Spanish (es), Hindi (hi), Nepali (ne), Modern Standard Arabic (msa), and Egyptian Arabic (arz). The scores for LID, NER, and POS are weighted F_1, micro F_1 with spans, and accuracy, respectively. State-of-the-art performance reached as of October 19th, 2020: `https://ritual.uh.edu/lince/leaderboard`. 111
8.6 The confusion matrix on the development set of the LID task for Spanish-English. The labels are lang1 (English), lang2 (Spanish), mixed (partially in both languages), ambiguous (either one or the other language), fw (a language different than lang1 and lang2), ne (named entities), other, and unk (unrecognizable words). 114
8.7 Statistics across the development sets that compare sequence lengths before and after subword tokenization. **Tokens** refers to the original length of the sequences as provided in the benchmark, while **Subwords** means the same sequence further tokenized with the BPE algorithm employed in multilingual BERT. 116

LIST OF FIGURES

1.1 Social media infographics. 2

2.1 High-level NER timeline of data sources (upper half) and models (lower half). 14

3.1 Examples from the CoNLL 2003 and WNUT 2017 datasets. 21
3.2 This is a stacked model that uses a multi-task learning (MTL) network as feature extractor. Once the MTL model is optimized, it transfers the learned features to a conditional random fields (CRF) classifier. 24

4.1 A dependency tree that shows how words relate among each other. 36
4.2 The overall model architecture. 40
4.3 A dependency tree (a) with its relative attention matrix A (b). 45
4.4 The global attention probabilities. The more highlighted the token is, the more probability mass it has. The labels and predictions (italics) appear below each word. 45
4.5 A comparison of the relative attention matrices between the BLSTM (a) and TLSTM (b) models for tweets from the WNUT-2016 dataset. The matrices contain row-wise probability distributions that exclude the main diagonal (from bottom-left to top-right). Both models correctly predict the entities *justin bieber* and *ELLEN*, but their patterns are different. Note that the TLSTM attention (b) highlights the words *justin bieber* while the BLSTM attention barely captures those words (a). Importantly, the dependency tree (c) used by the TLSTM model is a multi-rooted tree, which highly influences the short memory state of the TLSTM (i.e., the top nodes are the last nodes to be processed). 46
4.6 Average of hops to the most attended words in the validation set with respect to the entity tokens. For sequences, the hops are the number of tokens in between the entity and the most attended word. For trees, the hops are the number of nodes in the path that connects the entity and the most attended word. 49

5.1 Top 30 most frequent entity instances in the (unobserved) test set and their overlap with the train and development sets (observed). The subfigure **(a)** shows the most common location entities from the CoNLL-2003 dataset, while the subfigure **(b)** describes de geo-location class in WNUT-2016. The x-axis denotes the frequency of the entity instances. 53
5.2 Overlap percentage of entity instances from the test set against the train and development sets. The x-axis denotes the frequency of the entity instances in the test. For instance, the frequency threshold $\tau = 1$ accounts for all the test entities that appear at least once in the train set. 57
5.3 The BERT results on the observed subset for the CoNLL-2003 (left) and WNUT-2016 (right) datasets. The data points start at frequency one since this is the observed entity subset. 59

6.1 LID label distribution used in LinCE. While HIN-ENG and SPA-ENG have very few tokens for `unk` and `fw` (<1%), MSA-EA and NEP-ENG do not have occurrences of such labels. Also, with the exception of MSA-EA, all the partitions are proposed for LinCE as described in Section 6.2.5. 75
6.2 POS label distribution used in LinCE. The partitions for both datasets are proposed for LinCE as described in Section 6.2.5. Note that the labels `UNK`, `SCONJ`, `AUX`, `INTJ`, and `PUNCT` only appear in the SPA-ENG corpus, whereas `PRON_WH` is unique for HIN-ENG. 77
6.3 NER label distribution used in LinCE. All the datasets have the BIO scheme, but I only show the entity types for simplicity. Note that HIN-ENG only contains `PER`, `LOC`, and `ORG`. Also, with the exception of MSA-EA, all the other partitions are proposed for LinCE as described in Section 6.2.5. . . . 78
6.4 Label distribution of the sentiment analysis corpus used in LinCE. Note that this distribution differs from the origial dataset. 80

7.1 Examples of code-switched tweets and their translations from the CS LID corpora for Hindi-English, Nepali-English and Spanish-English. The subscript `ne` refers to named entities and `other` is used for punctuation, emojis or usernames. English text appears in *italics* and other languages are underlined. 85
7.2 A) The left figure shows the overall model architecture, which contains ELMo followed by BLSTM and CRF, and a secondary task with a softmax layer using a *simplified* LID label set. The largest box describes the components of ELMo, which include the enhanced character n-gram module proposed in this paper. B) The right figure describes in detail the enhanced character n-gram mechanism inside ELMo. The figure shows the ELMo convolutions of a word as input and a single vector representation as output. 89
7.3 Visualization of the tri-gram attention weights for the 2016 Spanish-English LID dataset. The boxes contain the tri-grams of the word below them along with the right (✓) or wrong (✗) predictions by the model. 95

8.1 A Hindi-English code-switched tweet. Hindi is transliterated to the Roman script (highlighted) instead of using Devanagari. Trans.: *Keep calm and mind your own business!!!* 97
8.2 A tokenization example that compares BPE and character tokenization given an input sentence. For BPE, only the word *nanotechnology* is split because it does not appear in the vocabulary. This word is broken down into pieces that only resemble the morpheme *-ology* (i.e., *nano-tech-nology*). Also, note that the resulting subwords do not use the word *technology* as a piece despite being present in the vocabulary. For characters, the tokenization process does not segment words into subword pieces; instead, it provides a list of character sequences where each sequence represents a word. 98
8.3 The char2subword module is trained using the subword embedding lookup table from a pre-trained language model (e.g., mBERT). I incorporate noise in every word at the character level with single-character operations. 104
8.4 An example of an input and output of the pre-training setting with a masked language modeling (MLM) objective at the character level. 105
8.5 The fine-tuning scenarios with the char2subword module. The mBERT model structure (left) is added for reference. The full mode (center) shows the char2subword module instead of the subword embedding table. The hybrid mode (right) represents both the subword embedding table and the char2subword module. The char2subword module is only used when the input word does not appear in the vocabulary as a whole. 106
8.6 Precision up to the 15-th top elements. (a) A comparison of the most important objective functions for the approximation phase. (b) A comparison of the best approximation model before and after the pre-training stage. . . . 108
8.7 Character-level attention for a Spanish-English tweet. The connections between words read from left to right, and they represent the probability mass distributed across the sentence for one word (i.e., a self-attention head). The right figure shows the attention distribution across the eight heads (i.e., from H0 to H7). Translation: *"Alright, otherwise you know the consequences!! Eh, haha."* 112

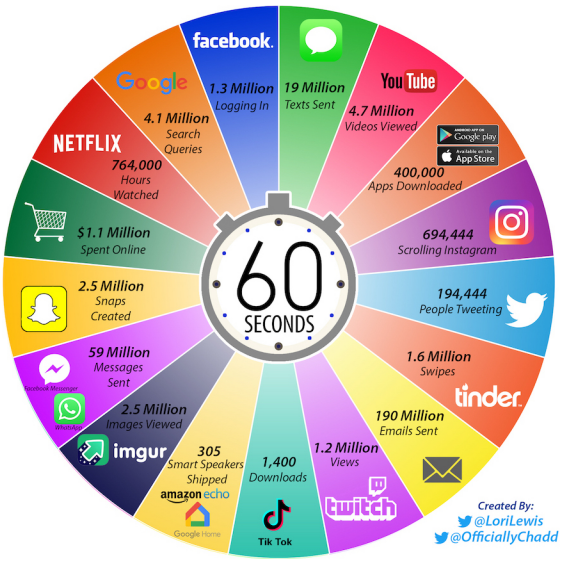
Chapter 1

Introduction

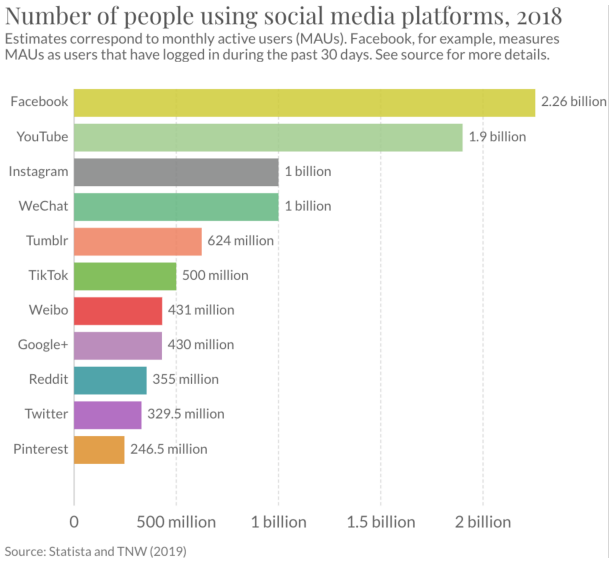
1.1 Motivation

Social media (SM) platforms have enabled people to communicate massively across the world. With such ability, there is an ever-growing flow of information that virtually reflects society every minute. Figure 1.1a shows a one-minute snapshot in 2020[1] of the most popular online applications, including SM activities of 59 million messages sent on WhatsApp and Facebook, 4.7 million videos viewed on YouTube, and 194 thousand people posting on Twitter. Moreover, behind this extensive amount of information, there is a large number of active users engaged in SM platforms. For instance, Facebook, the most popular platform, reached 2.8 billion monthly active users in 2018[2] (see Figure 1.1b). The vast number of users and the constantly generated content indisputably make SM an essential component of today's communication in society.

[1] `https://www.allaccess.com/merge/archive/31294/`, retrieved April 12th 2020.
[2] `https://ourworldindata.org/rise-of-social-media`, retrieved April 4th 2020.



(a) One minute in SM (2020).



(b) Monthly active users in SM (2018).

Figure 1.1: Social media infographics.

Social media is beyond online chatting. Under the social science umbrella, SM captures many aspects of human society and social relationships, ranging from marketing, economics, and politics to education, entertainment, and psychology. Consequently, SM platforms have become a window to study our society, where the relentless traffic of information has brought unprecedented opportunities to understand current worldwide events, human behavior, trends, and more.

As social media brings opportunities to understand our society, it also brings a variety of challenges to the automated processing of language. In particular, most of the textual content in SM is considered *noisy*; it does not always stick to the rules of the written language, and it tends to have misspellings, arbitrary abbreviations, orthographic inconsistencies, and flexible grammar. Although SM users easily adapt and conveniently use this behavior, the SM noise challenges conventional systems explicitly designed to perform on well-formatted and grammatically correct texts. Such systems need to normalize text to handle SM domains as they greatly depend on a predefined vocabulary. This preprocessing step is expensive to

maintain and requires many heuristics and hand-written rules to handle the diverse and arbitrary ways of writing in SM.

Additionally, SM platforms provide a unique space for multilingual content,[3] as they bring people together from all over the world. While this scenario demands user needs such as translating languages, it is often the case that the text contains more than one language, and even the alternation of languages in the same utterance (i.e., linguistic code-switching). This polyglot environment requires modern systems to adapt to a diverse range of languages, imposing another linguistic barrier to processing and understanding of text from SM domains.

Nevertheless, these linguistic challenges in SM are part of the continuous evolution of language. The grammatical rules, word meanings, orthography, regionalisms, and dialects have been changing across history and among communities from all over the world. In a globalized world, the back and forth interaction among communities has facilitated the evolution of language, eventually combining words and expressions from different cultures. SM has just sped up this process by vanishing language borders and bringing an amalgam of linguistic phenomena over the web.

This dissertation focuses on sequence labeling tasks in SM domains, covering challenges from noisy user-generated text and the linguistic code-switching phenomenon. I propose methods for three sequence labeling tasks: language identification (LID), named entity recognition (NER), and part-of-speech (POS) tagging. I demonstrate the effectiveness of adopting noise like phonological writing and flexible grammar as an informative feature for NER on English text. Additionally, I show that it is possible to adapt monolingual models to the code-switching setting using transfer learning, outperforming existing code-switching methods for LID, POS tagging, and NER.

[3]For instance, Facebook is constantly increasing the number of supported languages, currently covering more than 150 languages on the platform. Retrieved April 12th 2020 from `facebook.com/translations`.

1.2 Research Objectives

This dissertation focuses on low-level natural language processing, namely sequence labeling, for English and code-switched SM text. The general goal of this dissertation is the following:

> **General Goal:** *Improve the robustness of sequence labeling approaches for user-generated language in social media platforms.*

In particular, the first part of this dissertation concentrates on named entity recognition (NER) for English SM text. The studies in this part are motivated by the challenges that SM brings in terms of text deviating from standardized written language (e.g., slang, misspellings, flexible grammar, arbitrary abbreviations, etc.). The overall objective in this part is the following:

> **Objective 1**: *Model the user-generated noise as an intrinsic characteristic of the language in social media, rather than an aspect that needs to be removed or normalized.*

To this end, I propose linguistically-inspired methods for NER that exploit phonological writing and flexible syntax, two major aspects of the noise in SM text. Besides, given the specific case of NER, I investigate whether the state-of-the-art performance comes from memorization of entities or generalization of the entity contexts. This last study emphasizes the importance of assessing generalization beyond solely driving research progress based on an overall metric in a given dataset.

In the second part of this dissertation, I focus on three sequence labeling tasks for code-switched SM text: language identification, part-of-speech tagging, and named entity recognition. The studies in this part are motivated by the facts that i) collecting and labeling data for code-switching tasks is expensive and time-consuming, and ii) large pretrained language models capture multilingual knowledge from raw text using unsupervised learning.

Considering these aspects, the objective is the following:

> **Objective 2**: *Model linguistic code-switching for sequence labeling tasks effectively by leveraging monolingual and multilingual pre-trained knowledge from large language models.*

Specifically, I propose novel transfer learning methods from state-of-the-art monolingual and multilingual models, such as ELMo [87] and BERT [37], for the code-switching (CS) setting. These methods reduce the demand for CS annotated data and language-specific resources while exploiting already available monolingual and multilingual pre-trained knowledge.

The methods presented in this dissertation are meant to benefit higher-level NLP applications oriented to social media domains, including but not limited to question-answering, conversational systems, and information extraction.

1.3 Overview of Proposed Research

The main goal of this dissertation is to provide methods that handle sequence labeling tasks considering the social media challenges and the linguistic code-switching phenomenon. To achieve that, this dissertation is comprised of two main research lines: named entity recognition (Section 1.3.1), and sequence labeling tasks for code-switched data (Section 1.3.2).

1.3.1 Named Entity Recognition

In the first part of this dissertation, I study the named entity recognition (NER) task in social media domains. Specifically, I cover the following ideas:

1. **Modeling sound-driven writing (Chapter 3)**. SM noise in text refers to the deviation of standard writing. While the SM language seemingly behaves chaotically

and arbitrary, there is a clear pattern behind this writing. Users tend to type using abbreviations that emulate the way their messages would sound in actual speech. This sound-driven behavior allows them to type fast and conveniently without undermining the communication to their audience. By modeling phonological writing, it is possible to take advantage of a clean normalization effect on the data.

2. **Adapting to flexible syntax (Chapter 4)**. Despite the deviation of standard grammatical rules, SM text still follows some sense of grammar. By modeling the flexible syntax in SM text, it is possible to capture patterns behind the sentence formation, and hence, the interaction of entities with key words in the text. For instance, a dependency tree can connect a named entity syntactically with the main verb of the phrase. This allows making inferences about the entity judging based on the verbs that they are connected, which often reveals information about the entity types.

3. **Evaluating memorization and generalization (Chapter 5)**. Large pre-trained language models have driven recent advances in NLP, including NER. With such a large number of parameters it is possible that these models are substantially relying on memorization, rather than increasing their generalization capabilities. I investigate whether this is the case by studying standard NER datasets from both news and SM and the behavior of the models in these datasets.

1.3.2 Linguistic Code-Switching

In the second part of this dissertation, I focus on linguistic code-switching (CS), where I consider language identification, part-of-speech tagging, and named entity recognition as the sequence labeling tasks. Specifically, I cover the following aspects and ideas:

1. **Providing a centralized CS evaluation benchmark (Chapter 6)**. Despite that

many code-switching datasets have been released lately, there is no official way to determine which models perform the best across them. Even if researchers provide language-independent models, there is no standard centralized benchmark to compare across the code-switching community. I dedicate part of this dissertation to provide both a code-switching corpus for NER and a centralized benchmark across sequence labeling tasks and multiple language pairs.

2. **Transfer learning from English language models to CS (Chapter 7)**. CS is a global phenomenon that involves many languages all over the world. However, it is usually studied by language pairs, and depending on the languages, it tends to have different behaviors (e.g., Spanish-English behaves differently than Hindi-English), which makes the data annotation process expensive and time-consuming. Given those constraints, I aim at adapting pre-trained English models to code-switched English-paired text for sequence labeling by using the language identification task. This adaption proves beneficial for downstream NLP tasks such as named entity recognition and part-of-speech tagging.

3. **Adapting multilingual pre-trained language models to CS (Chapter 8)**. CS is inherently a multilingual phenomenon, which demands models that rely on multilingual pre-trained knowledge. This study aims at adapting large multilingual language models, such as BERT [37], to the code-switching setting. The adaptation considers SM challenges by replacing the word-piece tokenization and embedding process for character-level word representations, similar to ELMo [87]. The adaptation is conducted using a novel knowledge distillation method, which reduces the word embedding parameters of the model while preserving its pre-trained knowledge.

1.4 Contributions

I summarize below the contributions of this dissertation in two major categories.

Within the scope of named entity recognition:

- A new method that empirically shows the effectiveness of exploiting phonological representations of text in social media. This method targets text that replaces standard spelling with a more practical, phonologically inspired orthography, largely addressing misspellings, inconsistent orthography, and arbitrary abbreviations. This method provides a new way to approach the noise in social media text.

- A new method that demonstrates the generalization capabilities of modeling dependency grammar, even with permissive/incorrect syntax. The method recursively exploits the child-parent relationships of words according to the dependency tree of a sentence. It shows that words such as verbs and prepositions can reveal the types of the entities, emphasizing generalization. This method can potentially help to adapt to new entities (e.g., new topics) based on its reliance on word interactions.

- The results of a study that investigates whether the performance improvements of current state-of-the-art models come from entity memorization or context generalization. The analysis will provide insights about the optimized models and the diversity of the datasets (e.g., the overlap of entities appearing in both training and evaluation). This study emphasizes the importance of generalization in SM domains.

Within the scope of sequence labeling for linguistic code-switching:

- A new benchmark for linguistic code-switching evaluation (LinCE) that centralizes research progress on code-switching tasks. The benchmark covers ten publicly available

datasets, four tasks, and four language pairs. LinCE provides new data splits using a comprehensive stratification method that solves crucial problems in the original partitions. LinCE will provide leaderboards in an online platform to facilitate comparison of different methods across different datasets.

- A Spanish-English code-switching corpus annotated with named entity recognition labels. This corpus has been incorporated into LinCE.

- A new method that adapts easily-accessible English knowledge to English-paired code-switched languages. This method takes advantage of preliminary English knowledge from large pre-trained language models to identify other languages whose morphological patterns are different from English. This adaptation also proves effective to downstream NLP tasks while it reduces the dependency on code-switching resources (e.g., language-specific embeddings).

- As an extension of the previous contribution, this dissertation also provides a practical method that adapts multilingual language models to code-switching. This method reduces the amount of annotated data for code-switching sequence labeling tasks, while also keeping the multilingual diversity. This method benefits low-resource languages for downstream NLP tasks beyond English-paired code-switched languages, such as the language pair Modern Standard Arabic-Egyptian Arabic.

1.5 Publications

The work described in this dissertation relates to the following peer-reviewed conference papers (in chronological order):

1. **Aguilar, G.**, López-Monroy, A. P., González, F. A., and Solorio, T. (2018). Modeling

Noisiness to Recognize Named Entities using Multitask Neural Networks on Social Media. In *Proceedings of NAACL-HLT 2018* [4].

2. **Aguilar, G.**, Ling, Y., Zhang, Y., Yao, B., Fan, X., and Guo, C. (2020). Knowledge Distillation from Internal Representations. In *Proceedings of AAAI 2020* [3].

3. **Aguilar, G.**, Kar, S., and Solorio, T. (2020). LinCE: A Centralized Benchmark for Linguistic Code-switching Evaluation. In *Proceedings of LREC 2020* [2].

4. **Aguilar, G.** and Solorio, T. (2020). From English to Code-Switching: Transfer Learning with Strong Morphological Clues. In *Proceedings of ACL 2020* [9].

The following workshop and pre-print papers are also related to this dissertation:

1. **Aguilar, G.**, Maharjan, S., López-Monroy, A. P., and Solorio, T. (2017). A Multi-task Approach for Named Entity Recognition on Social Media Data. In *Proceedings of The 3rd Workshop on Noisy User-generated Text, EMNLP 2017* [5].

2. **Aguilar, G.**, AlGhamdi, F., Soto, V., Diab, M., Hirschberg, J., and Solorio, T. (2018). NER on Code-switched Data: Overview of the CALCS 2018 Shared Task. *The 3rd Workshop on Computational Approaches to Linguistic Code-Switching, ACL 2018* [1].

3. **Aguilar, G.** and Solorio, T. (2019). Dependency-Aware Named Entity Recognition with Global and Relative Attentions. *arXiv preprint arXiv:1909.05166* [8].

4. Patwa, P.[4], **Aguilar, G.**[4], Kar, S., Pandey, S., Pykl, S., Garrette, D., Björn, G., Chakraborty, T., Solorio, T., Das, A. (2020). SemEval-2020 Task 9: Overview of Sentiment Analysis of Code-Mixed Tweets. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval), COLING 2020* [83].

[4]Equal contribution.

5. **Aguilar, G.**, McCann, B., Niu, T., Rajani, N., Keskar, N., and Solorio, T. (2020). Char2Subword: Extending the Subword Embedding Space from Pre-trained Models Using Robust Character Compositionality. *arXiv preprint arXiv:2010.12730* [6].

This last research paper is not related to this dissertation, but it has been part of my work during the course of the PhD:

1. **Aguilar, G.**, Rozgić, V., Wang, W., Wang, C. (2019). Multimodal and Multi-view Models for Emotion Recognition. In *Proceedings of ACL 2019* [7].

2. Kar, S., **Aguilar, G.**, Lapata, M., Solorio, T. (2020). Multi-view Story Characterization from Movie Plot Synopses and Reviews. In *Proceedings of EMNLP 2020* [58].

3. Chen, S., **Aguilar, G.**, Neves, L., Solorio, T. (2020). A Caption Is Worth A Thousand Images: Investigating Image Captions for Multimodal Named Entity Recognition *arXiv preprint arXiv:2010.12712* [25].

Chapter 2

Literature Review

Many natural language processing (NLP) applications rely on the ability to extract low-level linguistic information from raw text. This information varies depending on the application needs, but it usually considers linguistic properties such as part-of-speech tags, named entities, and even language identification. The extraction of such information is conducted at either the token or token-span level, known as sequence labeling.

Social media has brought new challenges to sequence labeling systems that have been designed for standardized text. These systems sensibly drop in performance when they are moved from news to social media domains. However, the gap in performance drastically increases in the case of named entity recognition (NER), falling from around 90% to 40% absolute points on the F_1 score. Considering that NER receives the most significant impact in performance among sequence labeling tasks, I dedicate the first part of this chapter —and the first part of this dissertation– to the discussion NER. In the second part, I provide an overview of sequence labeling tasks under the perspective of linguistic code-switching.

2.1 Named Entity Recognition

2.1.1 Hand-Crafted Features and Rule-based Systems

In its early years, NER systems focused on newswire text, where the goal was to identify three types of entities: *person*, *corporation*, and *location*. These entity types were proposed in the MUC-6[1] [47] and MUC-7 [27] competitions. Most of the systems were based on heavily hand-crafted features and manually elaborated rules [23]. Nonetheless, maximum entropy for named entities was introduced by [76], and [23]. These approaches had many limitations due to the lack of learning capabilities and the reliance on manual features and rules.

A few years later, many researchers incorporated machine learning algorithms into their systems, but there was still a strong dependency on external resources and domain-specific features. For instance, in the 2003 CoNLL[2] shared task [113], the participants used gazetteers and other resources besides the ones provided by the organizers. In addition, the majority of the systems used maximum entropy [19, 26, 31, 42, 62] and hidden Markov models (HMMs) [42, 62, 74, 120]. The results of the participants move around 90% of the F_1 score. Similarly, [75] used a conditional random field (CRF) combined with web-augmented lexicons. The features were selected by hand-crafted rules and refined based on their relevance to the domain of the entities. They reduced the feature dependencies while still reaching a decent F_1 score of 80%. Moreover, [80] exploited Wikipedia resources taking advantage of structured data and reducing the human-annotated labels on the data. In general, the results of the systems were reasonable, yet the scalability and the extensive and expensive detailed rules were not; their methods were difficult to maintain and adapt to other domains where different rules were needed.

[1] MUC: Message Understanding Conference.
[2] CoNLL: Conference on Computational Natural Language Learning.

2.1.2 Adapting NER Systems to Social Media Text

While previous traditional ML approaches were considered feasible and effective for formal text, NER started to change into a different and more challenging task (see timeline in Figure 2.1). First, the formal settings that made the task manageable significantly changed to a noisy environment (i.e., social media), where the text hardly follows proper grammatical structures and frequently presents spelling inconsistencies and arbitrary abbreviations [94]. Second, the expansion to a bigger diversity of entity types established a harder task, where entities such as *movies*, *books*, *songs*, *products*, etc. are more heterogeneous than the conventional *person*, *corporation*, and *location* types [90, 66, 94, 35].

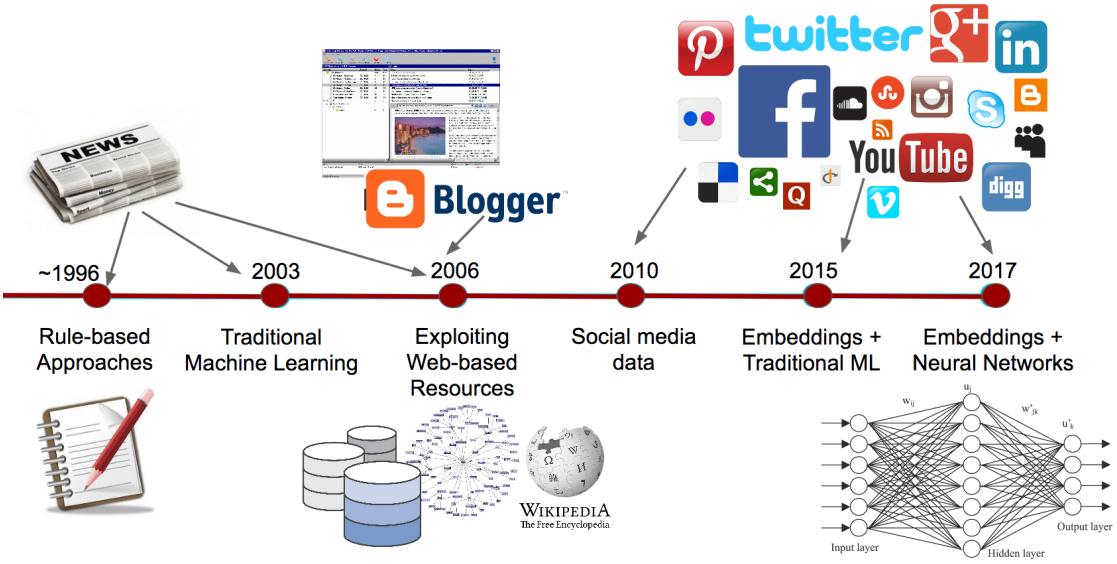


Figure 2.1: High-level NER timeline of data sources (upper half) and models (lower half).

The first approaches to these challenges were based on domain adaptation of the tools available at the time. For instance, [35] evaluated multiple NER tools in noisy environments: Stanford NER [41], ANNIE [22], among others. They reported that the majority of the tools are not capable of adapting to the noisy conditions. Additionally, they showed a significant

drop in performance when those tools are moved from formal text to noisy environments: a loss of 40% absolute points on the F_1 score.

2.1.3 Neural Networks for NER

Based on the success of word embeddings in 2013 [77], some researchers tried to learn such representations from social media text. For instance, [17] organized a NER competition at the 1st Workshop on Noisy User-generated Text (WNUT), where three of the participants used word embedding as features to train their machine learning models. Although the results were not as high as in formal text (see Table 2.1), pre-trained word embeddings allowed the participants to rely less on hand-crafted features [46].

Table 2.1: Results of different NER competitions. The performance degrades as the systems are moved to social media (SM) environments. The last row considers multiple SM domains, such as Twitter, YouTube, Reddit, and StackExchange.

Organizers	**Year**	**Competition**	**Domain**	**F_1 Score**	**Classes**
[47]	1996	MUC-6	Newswire	96.49%	2
[113]	2003	CoNLL	Newswire	88.76%	4
[110]	2016	WNUT	Twitter	52.41%	10
[36]	2017	WNUT	SM domains	41.86%	6

In 2016, different approaches emerged based on word embedding representations and deep learning models. For instance, [63] proposed a novel neural architecture model: a Bidirectional Long Short-Term Memory (BLSTM) with a CRF at the output layer. A better version of this architecture was proposed by [28], who used a BLSTM in conjunction with a Convolutional Neural Network (CNN). Likewise, [70] proposed a BLSTM and CNN as an end-to-end model, including the CRF loss function in the training process of their network. Importantly, however, these systems did not evaluate in the SM domain except for [84]. Instead, they benchmarked their systems on the CoNLL 2003 data, which belongs to the news domain. Following this line, the use of neural networks dominated subsequent

versions of the WNUT shared task [110, 36]. For instance, the winners of the 2016 edition, [64], have in common a CNN+BLSTM that passes the extracted features to a CRF classifier on the output layer. This has remained as the standard approach for NER.

2.1.4 Pre-trained Language Models for NER

More recently, the entire NLP community has been revolutionized by deep contextualized word embeddings based on language models, and the NER task is not the exception. For instance, [87] introduced ELMo, which uses bidirectional language models to predict the next token. The next token probability is modeled by multiple bidirectional LSTMs. After training their model, they transferred the learning and fine-tuned their original model (i.e., they adapted the last layer) to outperform a wide variety of NLP tasks, including NER. Similarly, [11] used a language model but only relying on characters: this accelerated the training phase and also provided some improvements on the NER CoNLL 2003 dataset. Finally, the NLP community has seen more approaches using language models that are substantially larger neural networks than before [54, 37]. Research has shown that these models capture many linguistic aspects, which make them very convenient for other tasks [87, 88, 124].

2.2 Linguistic Code-Switching

Linguistic code-switching (CS) has been studied in the context of many NLP tasks [105], including language identification [106, 18], part-of-speech tagging [109, 108, 78, 32, 107], named entity recognition [1], parsing [82], sentiment analysis [117], and question answering [92, 24]. Many code-switching datasets have been made available through the shared-task

series FIRE[3] [102, 29, 96] and CALCS[4] [106, 78, 1], which have focused mostly on core NLP tasks. Additionally, other researchers have provided datasets for dialect recognition [49], humor detection [60], sub-word code-switching detection [71], among others. While it is important to acknowledge the availability and recent growth of datasets, I narrow the discussion to sequence labeling tasks.

In the case of language identification (LID) at the token level, researchers have evaluated approaches such as conditional random fields (CRF) with hand-crafted features [12], LSTM models with word and character embeddings [73, 98], code-mixed word embeddings [91], and transfer learning [9]. While most of these approaches reach over 90% of accuracy regardless of the language pairs, it is hard to determine which model is the best overall and what the trade-offs are by using one instead of the others.

Likewise, for part-of-speech (POS) tagging, the community has explored tools that heavily rely on monolingual hand-crafted linguistic information and morphological features [13], traditional ML techniques (e.g., SVM) with heuristics that exploit monolingual resources [107], combined monolingual taggers, including CRF and Random Forest [56], and jointly modeling POS tagging with LID using recurrent neural networks [109]. Although such approaches are effective on their datasets at hand, they are language-specific and not easy to compare across each other.

A slightly different trend has been marked in named entity recognition (NER). Although the main problem in NER has been the lack of datasets, it is until recently that researchers have provided a few corpora on Hindi-English [104], Spanish-English and Modern Standard Arabic-Egyptian Arabic [1]. The participants of the 2018 CALCS competition proposed models based on standard neural NER architectures (e.g., character CNN, followed by a word-based LSTM, and CRF) [45], including variations with attention [119] and multi-task

[3]Forum for Information Retrieval Evaluation.
[4]Computational Approaches to Linguistic Code-Switching.

learning [115]. Additionally, most of the participants exploited publicly available resources such as gazetteers as well as monolingual and multilingual embeddings [122]. While the CALCS competition provided datasets on Spanish-English and Modern Standard Arabic-Egyptian Arabic simultaneously, the participants were allowed to provide predictions on one or both competitions. This flexibility left the question open regarding which model was overall the best across language pairs.

Although there has been significant progress in code-switching overall, CS still lacks advancements in many NLP tasks. Additionally, CS tends to advance guided by language-specific challenges, usually providing sparse technologies that may not necessarily be effective for other language pairs. This dissertation takes this scenario as an opportunity to contribute to a consolidated benchmark while also providing a unified framework for code-switching settings.

2.3 The Research Direction

The recent advances in NLP have brought significant improvements for sequence labeling tasks [87, 37] as well as higher NLP applications. However, these advances primarily rely on word embeddings and language models that are pre-trained on a large amount of text, excluding social media data. As a result, models tend to perform poorly on social media data since there is a need for adapting models to the new domain. Although it is possible to pre-train such models on social media text [11], the language in social media evolves fast, demanding practical approaches that immediately adapt to new linguistic challenges. These challenges include phonological text, new vocabulary, language transliteration, and linguistic code-switching. This dissertation aims at providing methods that can satisfy the fast-paced evolution of language in social media.

Part I

Named Entity Recognition on Social Media Text

Chapter 3

Modeling Social Media Noise

One of the core tasks in natural language processing (NLP) is named entity recognition (NER). NER is a sequence labeling task that consists of selecting the words that describe entities and recognizing their types (e.g., a person, location, company, etc.). Recognizing entities in the text is typically one of the first tasks in the pipeline of many NLP applications, including machine translation, summarization, sentiment analysis, and question answering.

Traditional machine learning systems have proven to be effective in standard text, where grammatical errors are minimal, and writers stick to the rules of the written language [42, 26]. However, those traditional systems dramatically fail on informal text, where improper grammatical structures, spelling inconsistencies, and slang prevail [94]. For instance, Table 2.1 (Chapter 2) shows a snapshot of NER systems' performances during the last years, where the results drop from 96.49% to 41.86% on the F_1 metric as the models move from formal to informal text. Although the results are not directly comparable because they consider different conditions and challenges, they serve as strong evidence that the NER task in social media is far from being solved.

Researchers have previously approached NER using different neural network architectures. For instance, [28] proposed a neural model using Convolutional Neural Networks

(CNN) for characters and a bidirectional Long Short Term Memory (LSTM) for words. Their model learned from word embeddings, capitalization, and lexicon features. On a slightly different approach, [63] used a BLSTM with conditional random fields (CRF) at the output layer, removing the dependencies on external resources. Moreover, [70] proposed an end-to-end BLSTM-CNN-CRF network, whose objective function is the negative log-likelihood loss determined by the CRF. These architectures were benchmarked on the standard CoNLL 2003 dataset [113]. Although most of the work has focused on formal datasets, similar approaches have been evaluated on SM domains [110, 36]. In the Workshop on Noisy User-generated Text (WNUT) 2016, [64], the winners of the social media NER shared task used a BLSTM-CRF model that induced features from an orthographic representation of the text. In the WNUT 2017 shared task, I proposed a multi-task learning (MTL) network that transferred the knowledge to a CRF classifier for the final prediction [5]. The method achieves first place in the competition.

This chapter focuses on addressing specific social media challenges for the NER task. I propose that, what is traditionally referred to as noise (i.e., misspellings, inconsistent orthography, emerging abbreviations, and slang), should be modeled *as is* since it is an inherent characteristic of SM text (see the examples in Figure 3.1).

CoNLL 2003 - News domain
[**Spanish**]$_{\text{MISC}}$ Farm Minister [**Loyola de Palacio**]$_{\text{PER}}$ had earlier accused [**Fischler**]$_{\text{PER}}$ at an [**EU**]$_{\text{ORG}}$ farm ministers ' meeting of causing unjustified alarm through " dangerous generalisation . "

WNUT 2017 - Twitter domain
been listenin to [**trey**]$_{\text{PER}}$ allll week ... can u luv someone u never met ?? bcuz i think im in luv yeeuuuuppp !!!

Figure 3.1: Examples from the CoNLL 2003 and WNUT 2017 datasets.

Specifically, the proposed model attempts to address i) misspellings using word- and

character-level representations, ii) grammatical mistakes with SM-oriented part-of-speech tags [81], iii) sound-driven text with phonetic and phonological features [20], and iv) the intrinsic skewness of NER datasets by applying class weights. It is worth noting that the model does not rely on capitalization or any external resources such as gazetteers. The reasons are that capitalization is arbitrarily used in SM environments, and gazetteers are expensive resources to develop for a scenario where novel entities constantly and rapidly emerge [36, 14].

3.1 Methodology

3.1.1 Feature Representation

Semantic features Semantic features are crucial for the model as they provide contextual information about the entities. I use the pre-trained word embedding model provided by [46]. This model has been trained on 1 million tweets (roughly 1% of the tweets in a year) with the skip-gram algorithm. These embeddings are also effective in other SM domains besides Twitter [5].

Syntactic features Syntactic features help the models deal with word disambiguation based on the grammatical role that the words play on a sentence. I capture grammatical patterns using the part-of-speech (POS) tagger provided by [81]. This POS tagger has custom labels suitable for SM data (i.e., the tagger considers emojis, hashtags, URLs, and others).

Phonetic and phonological features I also consider the phonetic and phonological aspects of the data at the character level. Table 3.1 shows an example of two phrases: the first sentence is taken from SM, and the second one is its normalized representation. Even

Table 3.1: Noisy and normalized text with equal International Phonetic Alphabet mappings.

Sentence	IPA
u hav to b KIDDDDING me	/ju hæv tə bi kɪdɪŋ mi/
you have to be kidding me	/ju hæv tə bi kɪdɪŋ mi/

though both phrases' spellings are significantly different, it is possible to map them to the same representation using their phonetics and phonological (articulatory) aspects. In other words, the assumption is that social media writers heavily rely on the way that words sound while they write. I use the Epitran[1] library [20], which transliterates graphemes to phonemes with the International Phonetic Alphabet (IPA). In addition to the IPA phonemes, I also use the phonological (articulatory) features generated by the PanPhon[2] library [79]. These features provide articulatory information such as how the mouth and nasal areas are involved in elaborating sounds while people speak.

3.1.2 Model Architecture

I propose a stacked model based on two phases: the feature extraction and the sequence labeling.[3] For the first phase, I define a multi-task neural network that acts as a word-level feature extractor. The idea is to leverage the context and linguistic properties in the input to extract fine-grained word-level features. The multi-task network uses two tasks:

- **Segmentation**. This task focuses on the Begin-Inside-Outside (BIO) scheme of the labels, also known as chunking. For a given NE, the model predicts whether a word is B, I, or O regardless of the entity type (e.g., person, location, etc.). This allows the model to learn how entities are treated in general, rather than associating the types to specific contexts.

[1]https://github.com/dmort27/epitran
[2]https://github.com/dmort27/panphon
[3]This model improves upon the system I presented in the WNUT shared task in 2017.

- **Categorization**. In this case, the model has to predict the types of entities along with the BIO scheme (e.g., B-person, I-person, etc.), which represent the final labels. Note that the model makes such predictions without taking into consideration the final sequence of labels.

For the second phase, I use the features extracted by the model to train a conditional random fields (CRF) classifier for the final sequence labeling. This phase aims to use hidden representations from the multi-task network to refine the final sequence of labels. Unlike the multi-task network, the CRF classifier looks at the full sequence to make its final prediction. The overall model is shown in Figure 3.2.

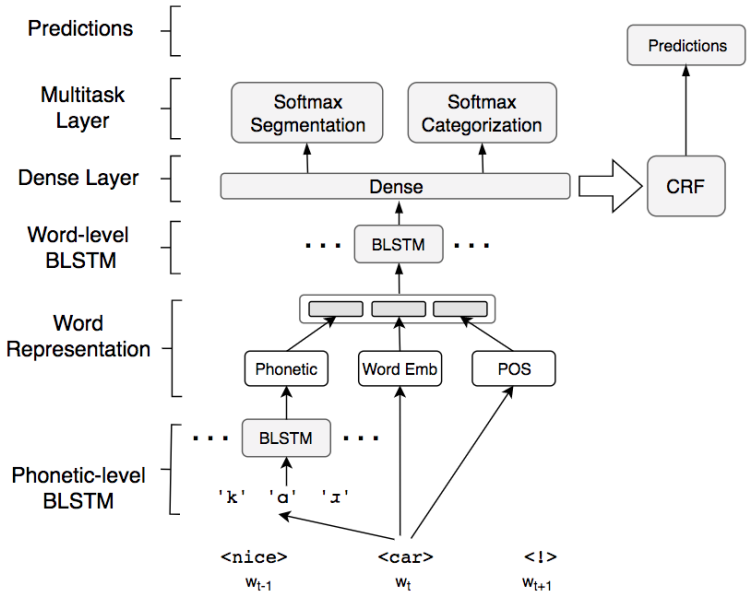


Figure 3.2: This is a stacked model that uses a multi-task learning (MTL) network as feature extractor. Once the MTL model is optimized, it transfers the learned features to a conditional random fields (CRF) classifier.

Multi-Task Learning Model Consider the sentence $X = [x_1, x_2, ..., x_n]$ where x_i is the i^{th} word in the sentence. I define the word embedding function $\alpha : \mathcal{V}_x \rightarrow \mathbb{R}^{dim_x}$ that maps each word x_i in the vocabulary $\mathcal{V}_x$ to the sequence of word embedding vectors $\mathbf{x} =$

$[\alpha(x_1), \ldots, \alpha(x_n)]$. Similarly, let $\beta : \mathcal{V}_p \rightarrow \mathbb{R}^{dim_p}$ be the POS tag embedding function that projects the sequence of POS tags into the sequence of vectors $\mathbf{p} = [\beta(p_1), \ldots, \beta(p_n)]$ For the phonetic segments of a word, I define a character-level function $\gamma : \mathcal{V}_q \rightarrow \mathbb{R}^{|\mathcal{V}_q|+dim_{PanPhon}}$ that maps each phonetic character to a one-hot vector of the International Phonetic Alphabet (i.e., $\mathcal{V}_q$) concatenated with the 21 (i.e., $dim_{PanPhon}$) phonological features of the PanPhon library (e.g., tongue position, movement of lips, etc.) [20]. Then, the phonetic and phonology embedding sequence for a given word is defined as $\mathbf{q} = [\gamma(q_1), ..., \gamma(q_m)]$. Notice that the embedding matrices $\mathbf{p}$ and $\mathbf{q}$ for the POS tags and phonetics are learned from scratch during training. The word embedding matrix $\mathbf{x}$ is initialized from pre-trained static embeddings.

I apply a character-level LSTM [53] to the $\mathbf{q}$ matrix on forward and backward directions. Then, I concatenate the output from both directions as follows:

$$\begin{aligned}
\overrightarrow{\mathbf{h}} &= \text{LSTM}(\{\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_m\}) \\
\overleftarrow{\mathbf{h}} &= \text{LSTM}(\{\mathbf{q}_m, \mathbf{q}_{m-1}, ..., \mathbf{q}_1\}) \\
\mathbf{h} &= [\overrightarrow{\mathbf{h}}; \overleftarrow{\mathbf{h}}]
\end{aligned}$$

This vector not only encodes the phonetic and phonological features, but it also captures morphological patterns at the character level based on the IPA representations. I concatenate the vector $\mathbf{h}$ with the word and POS tag representations: $\mathbf{a} = [\mathbf{x}_t; \mathbf{p}_t; \mathbf{h}_t]$ by their corresponding time step t in the sequence. I feed the $\mathbf{a}$ representation to a word-level bidirectional LSTM network [39], similar to the BLSTM described for the character level. The bidirectional LSTM generates a word-level representation that accounts for the sentence's context using semantics, syntax, phonetics, and phonological aspects. I linearly project the resulting sequence of vectors $\mathbf{a}$ to a fully-connected layer. Then, the final internal feature

representation $\mathbf{z}$ is obtained as follows:

$$\mathbf{r} = \text{BLSTM}(\{\mathbf{a}_1, \mathbf{a}_2, ...\mathbf{a}_n\}) \tag{3.1}$$

$$\mathbf{z}_i = \text{ReLU}(\mathbf{W}_a \mathbf{r}_i + \mathbf{b}) \tag{3.2}$$

Each of the internal representations $\mathbf{z}_i$ in the sequence is used as the input to the multi-task learning layers of the model. In particular, I define two objective functions that account for the segmentation and categorization tasks:

$$\mathcal{L}_{seg} = -\, y \log(softmax(\mathbf{W}_{seg}\ \mathbf{z} + \mathbf{b}))$$

$$\mathcal{L}_{cat} = -\, y \log(softmax(\mathbf{W}_{cat}\ \mathbf{z} + \mathbf{b}))$$

$$\mathcal{L} = \alpha \mathcal{L}_{seg} + \mathcal{L}_{cat}$$

The overall objective is the sum of both $\mathcal{L}_{seg}$ and $\mathcal{L}_{cat}$ with a weighting factor α that reduces the impact of the segmentation task (i.e., the categorization task is the ultimate goal, hence it needs to be prioritized).

Conditional Random Fields Once the MTL model is optimized, I use the feature representation $\mathbf{z}_i$ for every word as input to a conditional random fields (CRF) classifier. The idea of having these two phases separated is to let the MTL model focus on extracting dedicated representations for each word. This means that the MTL model's categorization predictions are not ideal for the sequence as a whole, although it is optimal for every word separately. Since the NER task is span-oriented (i.e., many words can form a single entity), the whole sequence must show consistency with the BIO scheme when detecting entities. The CRF classifier focuses on this task by choosing the optimal output sequence beyond only looking individually at each word.

3.2 Experimental Setting

3.2.1 Dataset

I focus this work on the WNUT 2017 dataset for NER [36]. This dataset covers multiple SM platforms and perfectly suits the purpose of this work. Table 3.2 shows the distribution of the dataset and its classes. The training set uses tweets, whereas the development set is based on YouTube comments. The testing set combines content from Reddit and StackExchange. The cross-domain nature of the dataset establishes an additional challenge to the task. For instance, besides the particularities of the domains (e.g., length of the sentences, domain-specific expressions such as hashtags, emojis, and others), the users tend to address different topics on each of the SM domains with varying levels of relaxed language and style [94, 110, 36]. Moreover, the predominant factors in those SM environments are the emerging and rare entities. As stated by [36], *emerging* describes the entity instances that started to appear in context recently (e.g., a movie title released a year ago), whereas *rare* depicts the entities that occur less than a certain number of times. It is worth noting that this dataset presents a significant challenge to systems that rely on external resources due to the rare and emerging properties.

Table 3.2: WNUT 2017 dataset with the class frequency distribution.

Classes	**Train**	**Development**	**Test**
person	995	46	532
location	793	238	188
group	414	64	202
creative-work	346	107	331
product	345	586	250
corporation	267	209	86
Classes	3,160	1,250	1,589

3.2.2 Implementation Details

I perform a straightforward pre-processing step on the data, which consists of replacing URLs, emojis, tags, and numbers with predefined tokens. Additionally, the pre-trained word embeddings' vocabulary is not sufficient to cover all the words in the WNUT dataset (i.e., training, validation, and testing sets have out-of-vocabulary words). I handle this situation using the Facebook library fastText [21]. This library can produce an embedding vector from the subword level of the word (i.e., n-grams). The advantage of fastText over other embedding learning algorithms is that it is possible to extract useful embeddings for OOV words by relying on the subword embedding space. For instance, if there is a missing letter in one word, the subword-level vector will be reasonably close to the correct spelling vector.

The model is trained using weighted classes, which forces the model to pay more attention to the less frequent labels. This is an essential step since the NE datasets usually show a skewed distribution, where the NE tokens represent approximately 10% of the entire corpus or less. Although weighing the classes improves the model's recall, the model is susceptible to this measure because it can lead to predicting entities even in cases where there are none. The weights were experimentally defined, keeping the same distribution but decreasing the loss on non-entity tokens.

Additionally, I define the model using the following hyperparameters: the phonetic and phonological BLSTM at the character level uses 64 units per direction, which adds up to 128 units. Similarly, the word level BLSTM uses 100 units per direction, which accounts for 200 units. The fully-connected layer has 100 neurons, and it uses a Rectified Linear Unit (ReLU) activation function. Also, I use a dropout operation before and after each BLSTM component. This forces the networks to find different paths to predict the data, which ultimately improves the generalization capabilities (i.e., they do not rely on a single path for specific inputs). The dropout probability value is 0.5. I train the model using the

Adam optimizer [61] with a learning rate of 0.001.

3.3 Results and Analysis

I provide the experimental results of the proposed model in Table 3.3. I experimented with two versions of the model, and I added the winning system results from the WNUT shared task for reference. The first version uses the two-phase training setting, which I refer to as the stacked model. The second version uses a single-phase training of the same model in an end-to-end (E2E) fashion (i.e., the CRF loss is back-propagated to the feature extractor). The results show that both versions, the end-to-end and stacked models, significantly outperform the state-of-the-art score by 2.28% and 3.69% F_1 points, respectively.

Table 3.3: The class-level and overall results of the systems on the WNUT 2017 dataset. **Stacked** means that the model was trained on a two-phase setting (i.e., first the MTL network, then the CRF classifier). **E2E** means that the model was trained in a single phase, back-propagating the gradients from the CRF down to the feature extractor. **WNUT** represents the winning system of the shared task by the UH-RiTUAL team in 2017. A t-test experiment shows that the stacked model improvement over the end-to-end model is statistically significant, with p-value < 0.0025 [38].

	Precision (%)			**Recall (%)**			**F_1 (%)**		
Classes	**Stacked**	**E2E**	**WNUT**	**Stacked**	**E2E**	**WNUT**	**Stacked**	**E2E**	**WNUT**
corporation	**33.33**	30.77	31.91	19.70	12.12	**22.73**	24.76	17.39	**26.55**
creative-work	50.00	**55.56**	36.67	**14.79**	10.56	7.75	**22.83**	17.75	12.79
group	47.76	**63.16**	41.79	**19.39**	14.55	16.97	**27.59**	23.65	24.14
location	62.20	**78.12**	56.92	**52.67**	50.00	49.33	57.04	**60.98**	52.86
person	**73.49**	71.15	70.72	51.05	**51.75**	50.12	**60.25**	59.92	58.66
product	**40.58**	34.29	30.77	**22.05**	9.45	9.45	**28.57**	14.81	14.46
Overall	61.06	**66.67**	57.54	**36.33**	32.99	32.90	**45.55**	44.14	41.86

The stacked model results predominantly outperform the scores of the WNUT system. The only class where the WNUT model performs better is *corporation*, even though it uses gazetteers across all classes. The entity diversity of the dataset can explain these results,

where the *emerging* and *rare* properties are difficult to capture with external resources. Nevertheless, the proposed model does not rely on similar external resources and still performs better.

Another important aspect is that the stacked system has lower precision than the end-to-end model, but its recall is the highest. This means that the stacked model is slightly better at generalizing than the other models since it can detect a more diverse set of entities. The surface form F_1 metric [36] supports that intuition as well. It assigns a better F_1 score to the stacked system (43.90%) than to the end-to-end model (42.79%) because the former finds more *rare* and *emerging* entities than the latter. Moreover, Table 3.3 also shows that the precision of the end-to-end model is higher than the rest of the systems. This tends to capture the most frequent entities and leave behind the *rare* ones, which explains the different behaviors between the precision and recall of both models.

MTL and CRF Comparison The feature extractor contains a category task that can produce predictions of the test set. I explored predicting the final labels with the feature extractor and compared the results against the CRF classifier's predictions. The CRF always outperformed the MTL network. For the best scores, the feature extractor achieved 40.64%, whereas the CRF reached 45.55%. This is consistent with previous research [63, 5] in that the label probabilities per individual word do not consider the whole sequence. Thus, a sequential algorithm such as a CRF can improve the results by learning global constraints of the BIO scheme (i.e., the *B-person* cannot be followed by *I-corporation*).

Ablation Experiment I explored the contribution of the features and different aspects of the model. For instance, I tried a BLSTM network using pre-trained word embeddings only. The results of this model set the baseline on 39.78% F_1 (see Table 3.4). This score is considerably close to the state-of-the-art performance, but improvements beyond that

are small. Table 3.4 shows an ablation experiment using the stacked model. The ablation reveals that weighting the classes is the most influential factor, which accounts for a 2.58% of F_1 score improvement. This aligns with the fact that the data is highly skewed, and thus, the model should pay more attention to the less frequent classes. The second most relevant aspect is the POS tags, which enhance the results by 1.10%. This improvement suggests that POS tags are important whether the dataset is from a noisy environment or not. Other researchers have found positive effects by using this feature on formal text [55]. Almost equally influential are the phonetic and phonological features that push the F_1 score by 0.93%. According to the ablation experiment, using phonetic and phonology along with the pre-trained word embeddings and POS tags can reach an F_1 measure of 41.81%, which is a very similar result to the state-of-the-art score, but with a simpler and more suitable model for SM environments (i.e., without gazetteers or capitalization).

Table 3.4: I performed an ablation experiment on the stacked model. The results in the table are the average of the scores of three iterations.

Model	**F_1**	**Delta**
Stacked Model	**45.55**	
- Multitask Learning	44.76	-0.79
- Character phonetics	43.83	-0.93
- Weighted classes	41.25	-2.58
- POS tag vectors	40.15	-1.10
- fastText OOV vectors	39.78	-0.37
- Pre-trained embeddings	12.72	-27.06

For the OOV problem, I used fastText to provide over 2,333 vectors for unknown words (around 13% of the vocabulary). However, the ablation experiment shows a small improvement when using fastText embeddings. This suggests that those word representations did not substantially contribute to the meaning of the context. Another aspect that I explored was adding all the letters of the dataset to the stacked model's character level without modifying the casing. Surprisingly, the models produced a slightly worse result (around -0.5%).

Intuitively, the character aspects are already captured by the model with the phonetic (IPA) representation, and the arbitrary use of capitalization renders this information useless. It is also worth noting that having phonetics instead of a language-dependent alphabet allows this approach's adaptability to other languages.

Multiple Tasks I explored the multi-task learning setting by empirically trying multiple combinations of auxiliary tasks. The best combination is the standard NER categorization along with the segmentation task. The segmentation slightly improves the binary task proposed by [5] by around 0.3%. Additionally, trying the binarization, segmentation, and categorization tasks together drops the results by about 0.2% for the categorization paired with the binary task. Moreover, the ablation experiment shows that the multi-task layer boosts the stacked model's performance with 0.79% of F_1 score.

Model Predictions Table 3.5 shows some predictions of the stacked model on the WNUT 2017 test set. In example number 1, the model can correctly label *Srinagar* as *person*, even though the model does not rely on gazetteers or capitalization. It is also important to mention that the word was not in the training or development set, which means that the network had to infer the entity purely from the context. Moreover, the second example shows that the model has problems to determine whether the article *the* belongs to an NE or not. This is an ambiguous problem that even humans struggle with. This example also has a variation on spelling for the words *Defence* and *Organisation*. I suspect that the mitigation of OOV words using the fastText library helped in this case. Also, from the phonetic perspective, the model treated the word *Defence* as if it was the word *Defense* because both words map to the same IPA sequence, /dɪfɛns/. In the third case, the model cannot identify the NE *Scout*, even though the context makes it reasonably easy.

Table 3.5: Model predictions of the Reddit text in WNUT 2017 dataset. The bold words are the gold labels, and the underlined words are the predictions.

No	Predictions
1	Road and airport closure isolate **Srinagar** as avalanche risk remains high
2	The **Defence Research Development Organisation** (**DRDO**) is working on four projects to develop new technologies for more accurate ...
3	Her name is **Scout** .

3.4 Limitations

Modeling noise is a challenging task due to its many forms in social media text (e.g., arbitrary abbreviations, regionalisms, new expressions, misspellings, etc.). Sound-driven writing is only one aspect, and even in the scope of this particular aspect, there are many more things to consider that go beyond this chapter. For instance, one of the limitations of the proposed method is that the model learns phonetic character-level features from scratch, restricting the model generalization to the aspects learned during a specific dataset training. In contrast with the pre-training and fine-tuning steps usually seen in current methods, the proposed model does not have a general sense of phonology and phonetics. Hence, the sound-driven writing abstraction of the model is limited to a shallow representation of the text.

Additionally, these representations are not learned from raw text. Instead, the model requires character-level inputs in the form of IPAs. While these IPAs are extracted automatically using external tools, they are not necessarily tailored to the social media domain. For instance, the tools are language-specific, and social media often presents multiple languages in the text. Nevertheless, showing the impact of modeling sound-driven writing motivates more linguistic-aware models (e.g., pre-training a language model that is aware of sound and then fine-tuning it on SM).

Another important aspect to note is the hard-to-tune multi-task setting. In this research, I tested different multi-task settings (binary, segmentation or chunking, and the main task),

and the results show that more tasks are not necessarily better. The complexity of choosing tasks is also tied to balancing each task loss's impact on the model. A model could generalize well for all the objectives (all the tasks) up to a point during training. It could then start optimizing more of the objectives that may not be the task of interest while having an overall lower error across tasks. That can be balanced with static or dynamic (i.e., trainable) weight factors per loss, but this experimentation goes beyond the point of modeling noise in social media texts.

3.5 Conclusion

I designed representations that are robust to SM data's inherent properties rather than focusing efforts on normalization. The proposed methods embrace challenges such as inconsistent spellings, diverse vocabulary, and sound-driven writing. Considering that SM is a prevalent communication channel that continuously generates massive amounts of data, it is practical to design NLP tools to process this domain *as is*. Additionally, I showed that the phonetic and phonological features are useful to capture sound-driven writing. This approach avoids the standard normalization process and boosts prediction performance. Furthermore, multi-task learning with segmentation and categorization is important to improve the models' results. Finally, the weighted classes force the model to pay more attention to skewed datasets. I showed that these components point to more suitable approaches for NER on social media data.

Chapter 4

Adapting to Flexible Syntax

In the study of the named entity recognition (NER) task, neural sequence labeling models have been vastly explored by the NLP community. Small variations aside, the majority of these models use a combination of a bidirectional LSTM and conditional random fields (CRF) to reach the state of the art performance [63, 70, 85, 57, 4]. Recently, transfer learning from pre-trained language models has played an important role on improving the performance even further [65, 87, 11, 37, 54]. However, these NER models mainly process the text as a straight sequence of words without explicitly considering the recursive nature of language.

Consider the phrase in Figure 4.1, "*Your friend Jason, who has been helping us, called you using his new Samsung.*" The main sentence is composed of a verb phrase whose head is *called*, which includes another verb phrase headed by the word *using*. Additionally, such recursive properties can produce more involved and longer sentences. This behavior can obscure relationships among words when the text is treated as a linear chain of tokens. In the same example, the words *Jason*, *called*, *using*, and *Samsung* easily describe the way in which the entities *Jason* and *Samsung* interact[1]. Nevertheless, existing models struggle to identify such relationships in long sentences, resulting in a drop in performance.

[1]Some verbs provide sufficient clues to determine the entity type of the subject that performs the action.

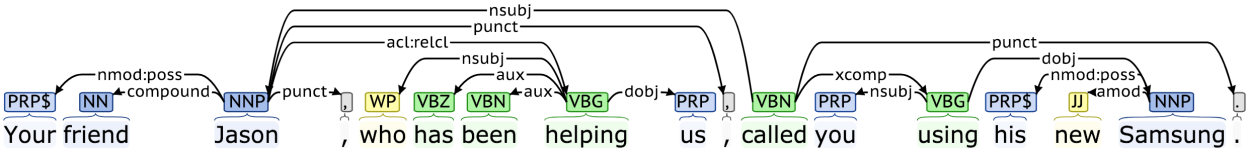


Figure 4.1: A dependency tree that shows how words relate among each other.

I propose a new approach for NER, where the goal is to enhance the syntactic relationships among words and combine such aspects with semantic representations commonly used in this task. The model extracts features from text using a Tree-LSTM [112] guided by dependency tree structures. As shown in Figure 4.1, the dependency trees connect the words based on the role they play in the sentence and the way they interact with each other. The output features are weighted with relative and global attention mechanisms. While the relative attention focuses on the most relevant words for the word being evaluated, the global attention spots the essential words over the whole sentence. After weighing the hidden vectors, I linearly project them into the tagging space. I predict the entity labels using a conditional random field classifier. The findings show that the model detects words that disclose the entity types based on their syntactic roles in a sentence (e.g., verbs such as *speak* and *write* are attended when the entity type is `PERSON`, whereas *meet* and *travel* strongly relate to `LOCATION`).

4.1 Methodology

4.1.1 Feature Representation

I represent the input data using words, part-of-speech tags, and dependency parses. For words , I employ deep contextualized representations using ELMo[2] [87]. ELMo provides vector representations that are entirely built out of characters. This allows overcoming the problem of out-of-vocabulary words by always having a vector based on morphological clues

[2] `https://allennlp.org/elmo`

for any given token. For POS tags and dependency relations, I use trainable embedding matrices that are optimized from scratch. POS tags have proven useful in previous research [28, 4], and dependency relations help the model to infer the interaction between nodes in the trees. Once I have a vector representation for every input feature, I concatenate them to form a single vector for every token in the sentence.

4.1.2 Model Architecture

Tree-LSTM The Tree-LSTM component, introduced as Child-Sum Tree-LSTM by [112], is a generalization of the standard LSTM cell [53] that can handle multiple inputs at every time step. Both cells are equivalent when the input tree is comprised of a single child at every level. The Child-Sum Tree-LSTM runs in a bottom-up fashion, which makes it equivalent to a reversed LSTM when the root is the first word in a straight sequence of tokens.

Relative attention Similar to the self-attention mechanism introduced by [116], I define an attention mechanism based on the scaled-dot product formulation. The relative attention is a particular case of the self-attention mechanism, where the main diagonal of the attention matrix is not taken into account to draw the probability distributions. More specifically, there is a probability distribution over all words w_j for every word w_i where $i \neq j$. The idea is to utilize the surrounding context for a word more than exploiting the word itself. The attention matrix is computed as

$$A = \text{ softmax}(d_a^{-0.5} Q K^T) \tag{4.1}$$

where $A \in \mathbb{R}^{N \times N}$ is a squared matrix that contains the attention weights for N words in a sentence, i and j denote the row and column indexes, such that $\sum_i^N \sum_{j \neq i}^N A_{ij} = N$. Q and K are linear transformations of the input using the query and key matrices $\mathrm{W}_Q \in \mathbb{R}^{d_a \times d_a}$

and $\mathrm{W}_K \in \mathbb{R}^{d_a \times d_a}$ where d_a is the dimension of the input and output matrices. The weighted values are calculated as follows:

$$M = \ AV + V \tag{4.2}$$

Similar to Q and K, V is a linear projection of the input using the value matrix $\mathrm{W}_V \in \mathbb{R}^{d_a \times d_a}$. Note that the matrix multiplication between A and V discards the words w_{ii} because A contains zeros in its main diagonal. Hence, I include this information by adding the matrix V to the product.[3]

Global attention In the case of the global attention, I use a fairly standard mechanism introduced by [16]. The idea is to concentrate mass probability over the words that capture the most relevant information across the sentence. Note that this mechanism conceptually differs from relative attention. While relative attention draws dedicated probability distributions conditioned on each word, the global attention is only conditioned by the context q, thus generated only once for the whole sentence. The attention mechanism uses the following equations:

$$\begin{aligned} u_i &= v^\intercal \tanh(\mathrm{W}_h h_i + b_h + \mathrm{W}_q q + b_q) \\ a_i &= \frac{\exp(u_i)}{\sum_{j=1}^{N} \exp(u_j)}, \quad \text{s.t.} \sum_{i=1}^{N} a_i = 1 \\ z_i &= a_i h_i \end{aligned} \tag{4.3}$$

where $\mathrm{W}_h \in \mathbb{R}^{d_a \times d_h}$, $\mathrm{W}_q \in \mathbb{R}^{d_a \times d_q}$, $b_h \in \mathbb{R}^{d_a}$, and $b_q \in \mathbb{R}^{d_a}$ are trainable parameters of the model. W_h and W_q are used to linearly project the hidden word vectors h_i and the query vector q into the attention space. In the case of using a Tree-LSTM, q is the root hidden vector h_{root}, whereas in a LSTM q is simply the last hidden state h_n. Finally, I multiply the

[3]This is equivalent to $M = (A + I)V$ where $I \in \mathbb{R}^{N \times N}$ is the identity matrix.

scalars a_i and their corresponding hidden vectors h_i to obtain the weighted sequence z.

Residual connections I incorporate residual connections [50] at every component of our module, followed by layer normalization as in [15]. The output of a given Sublayer is described as follows:

$$z = \text{LayerNorm}(x + \text{Sublayer}(x)) \tag{4.4}$$

where LayerNorm is an affine function that contains trainable parameters. Additionally, Sublayer can be any component of our model, such as a Tree-LSTM, relative attention, or global attention. I keep the same dimensions for inputs and outputs to simplify adding the vectors of any given module. This module only normalizes the output tensor in the last dimension.

Conditional random field I use a conditional random field (CRF) classifier at the top of our model to perform the sequential inference. The CRF takes vectors in the tagging space as input and produces the best sequence of labels using the Viterbi algorithm. CRF is well-known and widely used for sequence labeling because it learns the rules of transitioning from one label to another based on the feature vectors of the sequence as a whole instead of individually.

Consider the observation sequence of vectors $\mathbf{x} = [x_1, \ldots, x_n]$ and its corresponding target labels $\mathbf{y} = [y_1, \ldots, y_n]$. CRF computes the conditional probability of the target sequence $\mathbf{y}$ given the inputs $\mathbf{x}$ by globally normalizing the target score:

$$p_\theta(\mathbf{y}|\mathbf{x}) = \frac{1}{\mathbf{Z}_\theta(\mathbf{x})} \prod_{t=1}^{N} \psi(y_{t-1}, y_t, \mathbf{x}; \theta) \tag{4.5}$$

where $\mathbf{Z}_\theta(\mathbf{x})$ is a normalization term that adds up the products of $\psi(\cdot)$ for all the possible $\mathbf{y}$ sequences. $\psi(\cdot)$ is the potential parametric function that sums the transition and emission

features. I use the logistic expression of Equation 4.5 during training to optimize our model.

Overall architecture The overall model architecture is shown in Figure 4.2. I first embed the input sentence into a vector space using the token embedder module. This module concatenates the word, POS tag, and dependency relationship vectors into a single representation for every token. These vectors are fed into the semantic (on the left) and syntactic (on the right) feature extractors, which are stacked layers of bidirectional LSTMs and Tree-LSTMs, respectively. The model then concatenates the outputs of these components and feeds them into the relative and global attention modules.[4] The weighted vectors from the attention mechanisms are linearly projected into the tagging space and fed into a conditional random field classifier.

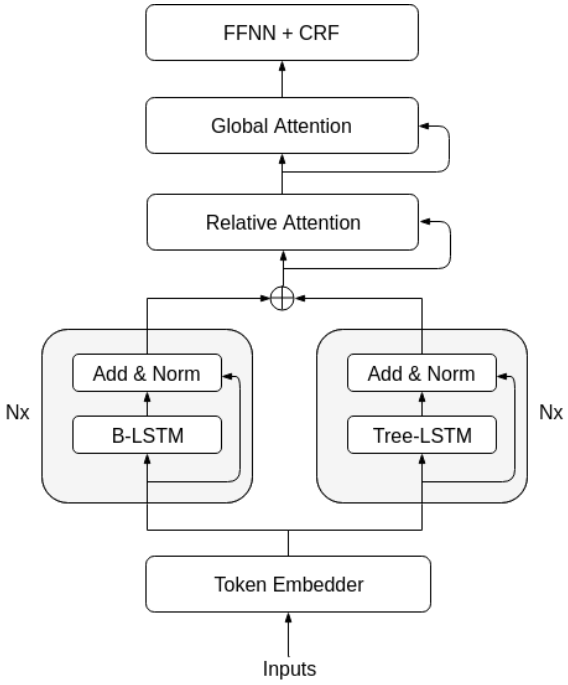


Figure 4.2: The overall model architecture.

I optimize the model by minimizing the negative log-likelihood loss produced by the CRF's forward algorithm. In addition to this loss, I include an ℓ_2 regularization term that

[4]The arrows that skip layers denote the residual connections.

targets the parameters of the semantic and syntactic feature extractors and the attention mechanisms. The idea is to reduce over-fitting when the model selects the features from the syntactic and semantic blocks and forces the attention mechanisms to avoid bias towards specific aspects of the sentences. Given the target sequence $\mathbf{y}$ and the predicted labels $\hat{\mathbf{y}}$, I define the objective function[5] of the model as follows:

$$\mathcal{L} = -\frac{1}{N}\sum_{i}^{N} y_i log(\hat{y}_i) + \lambda \sum_{k} w_k^2 \tag{4.6}$$

where N is the length of the sentence and w_k denotes the parameters of the feature extractors and the attention mechanisms. λ is the penalty that indicates how much of this regularization term will be added to the overall loss, and $log(\hat{y}_i)$ is determined by the logistic expression from Equation 4.5.

I reduce the loss of our model using Stochastic Gradient Descent (SGD) with momentum [111]. I train our models for 150 epochs and change the learning rate every epoch using cosine annealing and warm restarts [68]. Besides adding ℓ_2 regularization, I also prevent over-fitting by applying input variational dropout [43] to every component in our model.

4.2 Experiments and Results

Datasets I run experiments on CoNLL-2003 [113], SemEval-2010 Task 1 [93], WNUT-2016 [110], and WNUT-2017 [36]. Table 4.1 shows a summary of the data splits for each corpus along with the number of entities. I use datasets from the news and social media domains to compare the effectiveness of the method using dependency trees. The SemEval-2010 dataset contains dependency trees manually annotated. For the CoNLL-2003 and WNUT datasets, I auto-generated dependency parsing using off-the-shelf methods. Note that the social media

[5]This function expresses the loss for a single input sequence.

Table 4.1: The data splits on each corpus along with the number of named entities.

Corpus	Domain	Train	Development	Test	NEs
CoNLL 2003	News	14,041	3,250	3,453	4
SemEval 2010	News	3,648	741	1,141	22
WNUT 2016	Social Media	2,394	1,000	3,850	10
WNUT 2017	Social Media	3,394	1,009	1,287	6

dependency trees are noisier than the ones for the news datasets. Therefore, each dataset provides a level of reliability for the dependency trees, being SemEval-2010 the most reliable and WNUT-2016 and 2017 the least reliable.

Experiments The goal of the experimentation is to compare the performance between sequential processing (i.e., BLSTM) vs. hierarchical processing guided by dependency trees (i.e., TLSTM). Hence, the experiments on Table 4.2 are layout in row pairs for each additional component or input feature added to the processing methods.

For experiments 1.1 to 1.4, adding POS tags improves significantly the performance compared to only using the words in either the BLSTM or TLSTM models. The benefit of adding POS tags as input is consistent with previous research (see Section 4.1.1).

In general, the results show that the TLSTM works better than the BLSTM when the dependency trees are reliable. The SemEval-2010 dataset provides manually-annotated trees, while the other datasets use trees automatically generated. Common problems for auto-generated trees are nodes mistakenly connected within the tree, as well as multi-rooted sentences (i.e., multiple trees for a single sentence). The former issue can tangle the word relationships while processing the word-to-word connections (i.e., parent and children) across the sentence tree. The latter issue prevents the information to flow from one tree to another, hence restricting the context of one tree to the rest of the text. This scenario is more prone to happen on social media text, where multiple utterances are combined within a single post.

Table 4.2: The F_1 scores on the validation sets. BLSTM and TLSTM are **b**idirectional and **t**ree LSTMs. † denotes residual connections on every component. RA and GA refer to relative and global attentions. All the experiments use ELMo embeddings and CRF. Running t-test shows that the experiment numbers between methods (BLSTM vs. TLSTM) are statistical significant with all p-value < 0.01. The experiment 3.1 on the SemEval-2010 dataset is also statistically significant against the previous best score given by experiment 2.6.

Experiment	**CoNLL-03**	**SemEval-10**	**WNUT-16**	**WNUT-17**
Baseline models				
1.1. BLSTM$_{\text{WORD}}$	**89.06**	**80.81**	46.89	**58.63**
1.2. TLSTM$_{\text{WORD}}$	86.72	79.53	**47.32**	57.13
1.3. BLSTM$_{\text{WORD+POS}}$	**91.22**	80.31	47.22	**59.04**
1.4. TLSTM$_{\text{WORD+POS}}$	88.91	**84.07**	**47.54**	57.87
Attention modules				
2.1. BLSTM + RA†	**91.65**	84.19	47.52	57.46
2.2. TLSTM + RA†	91.13	**84.94**	**47.78**	**58.31**
2.4. BLSTM + GA†	**91.39**	84.87	**47.87**	**58.82**
2.3. TLSTM + GA†	90.62	**85.09**	47.25	57.09
2.5. BLSTM + RA + GA†	91.63	85.12	**48.67**	**59.28**
2.6. TLSTM + RA + GA†	90.60	**86.24**	47.96	57.42
Combining feature extractors				
3.1. TLSTM + BLSTM + RA + GA†	91.17	86.49	45.35	55.19

For the WNUT-2016 and -2017 datasets, I generate the trees with the TweeboParser tool [81]. This tool is tailored to Twitter data, which is fully compatible with the WNUT-2016 dataset. However, the WNUT-2017 uses Twitter for the training set, YouTube comments for the development set, and StackExchange posts for the test set. This domain shift is not ideal for the automatically-generated trees, which explains the lower performance of experiments 1.2 and 1.4 compared to the experiments 1.1 and 1.3 on WNUT-2017. Additionally, the WNUT-2016 experiments favor less the BLSTM scores; the TLSTM is more competitive to BLSTM than in WNUT-2017 dataset. Nevertheless, having multiple trees per tweet negatively impacts the TLSTM model.

Importantly, the relative attention helps to reduce the problem of multiple trees and ambiguous node connections in the dependency parse. For instance, experiments 2.2 improve

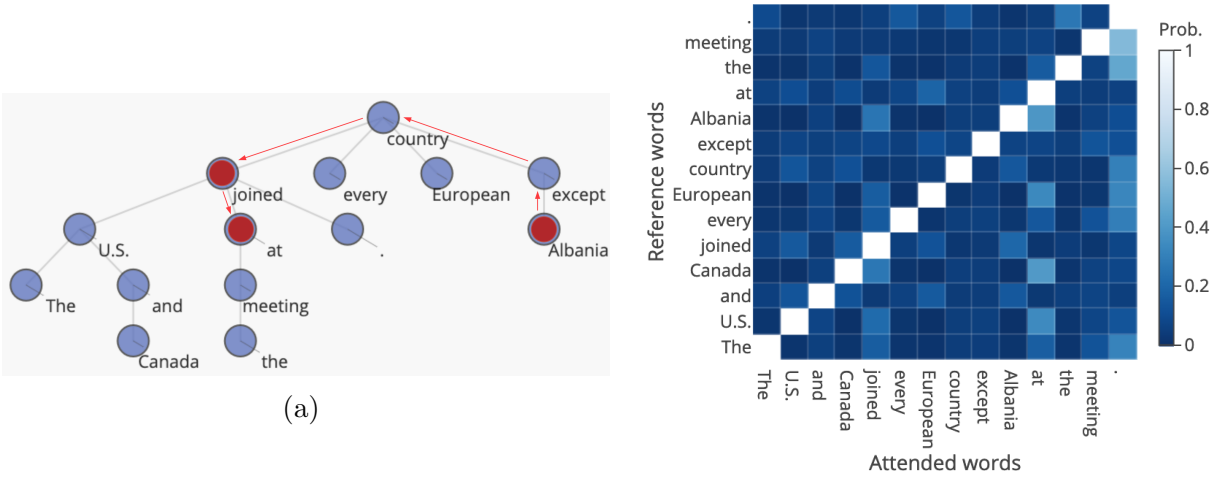
their performance from experiment 1.2 across datasets. Intuitively, the relative attention helps to reconnect the isolated trees by having a Cartesian product from the self-attention mechanism (i.e., one word attending all other words in the sentence).

Finally, the experiment 3.1 shows that combining all the components as well as the *LSTM models can yield good results if the dependency trees are reliable (experiment 3.1 on the SemEval-2010 dataset). Otherwise, the TLSTM can have no positive effect, or even harm the results.

4.3 Analysis

Relative attention Figure 4.3b shows the relative attention matrix for the dependency tree in Figure 4.3a. Each row of the matrix draws a probability distribution over the rest of the words (columns). By inspecting the matrix, it is easy to note that the words *at* and *joined* in the x-axis are the most relevant for *Albania* in the y-axis, whereas *U.S.*, *Canada*, *European*, and *Albania* (x-axis) are the most important for *joined* (y-axis). This supports the idea that prioritizing words based on their relationships can provide different perspectives of the sentence at the word level. Additionally, when I inspect the models BLSTM and TLSTM separately, I find different patterns captured in their attention matrices; for BLSTM the relations are prioritized semantically (e.g., verbs are the most important parts and prepositions are hardly highlighted), whereas for TLSTM it is more syntactically (e.g., prepositions, verbs, punctuation are all relevant). Merging both techniques improves the results because they are complementary.

Global attention Figure 4.4 shows the attended words once the relative attention matrix has extracted the prioritized relations. The figure shows a combination of both syntactic and semantic patterns. That is, while words like *joined* and *meeting* would be semantically



(a) (b)

Figure 4.3: A dependency tree (a) with its relative attention matrix A (b).

expected, the model also focus on *except*, *at*, and the punctuation. Importantly, the model does not act as an entity spotter at this level.[6] Instead, it relies on the syntactic structure of the sentence. Additionally, note that the word *meeting* was not too relevant in the relative attention matrix, but at the global perspective, it becomes important. This suggests that the attention mechanisms are complementary.

The	U.S.	and	Canada	joined	every	European	country	except	Albania	at	the	meeting	.
O	B-GPE	O	B-GPE	O	O	B-NORP	O	O	B-GPE	O	O	O	O
O	*B-GPE*	*O*	*B-GPE*	*O*	*O*	*B-NORP*	*O*	*O*	*B-GPE*	*O*	*O*	*O*	*O*

Figure 4.4: The global attention probabilities. The more highlighted the token is, the more probability mass it has. The labels and predictions (italics) appear below each word.

Blstm vs. Tlstm Figure 4.5 shows the relative attention probabilities using the BLSTM (Figure 4.5a) and TLSTM (Figure 4.5b) models. For Figure 4.5a, the probability mass tends to concentrate on the last words of the sentence (e.g., *will cry for days*), while Figure 4.5b shows that the attention mass spreads across the sentence with clear emphasis on entity words like *justin bieber* and *ELLEN*. Despite concentrating the probability mass over the entity

[6] e.g., $\text{BLSTM}_{\text{WORD}}$ with global attention tends to highlight entities suggesting memorization rather than generalization

words, it is worth noting that the relative attention ignores the i-th word when computing the probabilities for the same word w_i. This forces the model to look at the context rather than just memorizing the entity every time.

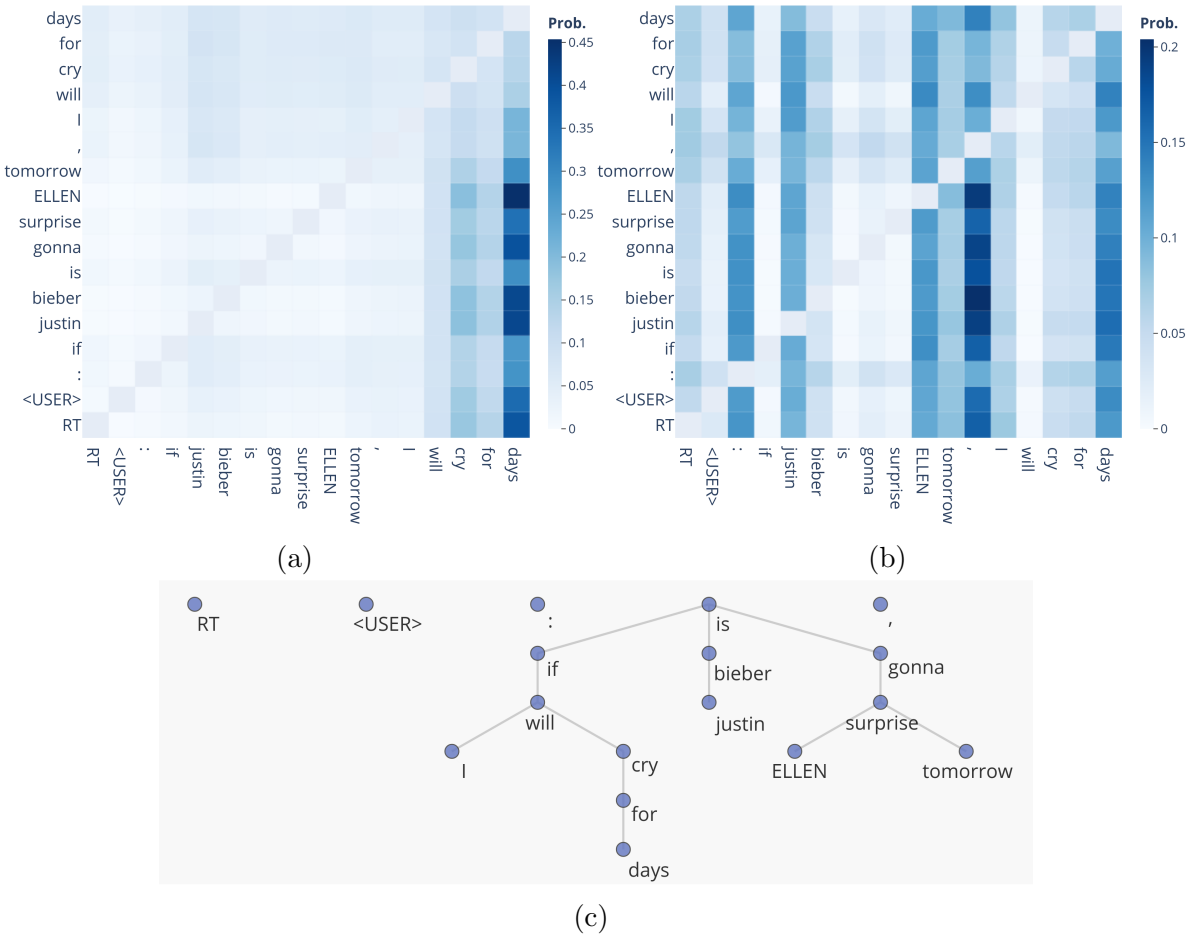


Figure 4.5: A comparison of the relative attention matrices between the BLSTM (a) and TLSTM (b) models for tweets from the WNUT-2016 dataset. The matrices contain row-wise probability distributions that exclude the main diagonal (from bottom-left to top-right). Both models correctly predict the entities *justin bieber* and *ELLEN*, but their patterns are different. Note that the TLSTM attention (b) highlights the words *justin bieber* while the BLSTM attention barely captures those words (a). Importantly, the dependency tree (c) used by the TLSTM model is a multi-rooted tree, which highly influences the short memory state of the TLSTM (i.e., the top nodes are the last nodes to be processed).

Additionally, the dependency tree used by the TLSTM model contains multiple roots (e.g., *RT*, *<USER>*, and punctuation). These roots are not connected by the TLSTM model

because it requires a multi-root node as input in addition to the children states. This means that the information never flows from one tree to another, completely isolating the context from one or more fragments of the text.[7] The attention matrix in Figure 4.5b shows that the isolated roots attract mass probability from the attention.

Entity connections I investigate whether the attended words with respect to the entity tokens follow any particular pattern. I conduct the study by controlling over the entity types and their corresponding POS tags. Specifically, I only consider the entity tokens whose POS tags are nouns (e.g., `NN`, `NNS`, `NNP`, or `NNPS`). Then, I extract the top three most attended words along with their POS tags from the SemEval 2010 validation set. In Table 4.3, I show the coverage of the most attended POS tags and their corresponding words for `PERSON` and `LOCATION`. For the type `PERSON`, the nouns *president*, *assistant*, and *officer* are roles that only people perform, which easily discriminate the type `PERSON` from any other entity type. In the case of its attended verbs, *speak*, *love*, and *die* are actions also performed by people (i.e., the entity is the subject of the sentence). For the type `LOCATION`, the verbs *entitle*, *travel*, and *meet* appear in cases where the location is the object of the sentence, and as such, these verbs disclose enough information to recognize that the entity is a place. Similarly, the nouns *pollution*, *earthquake*, and *temperatures* are commonly used to describe the state or events in a specific location. These findings are consistent with the initial intuition that specific words, along with their syntactic roles, can disclose important clues to recognize a given entity.

Attended word distances. I calculate the distance between the entities and its most attended words (i.e., the words with the highest probabilities). I take the attention matrix A and look for the rows whose tokens are labeled as entities. Then, I extract the most

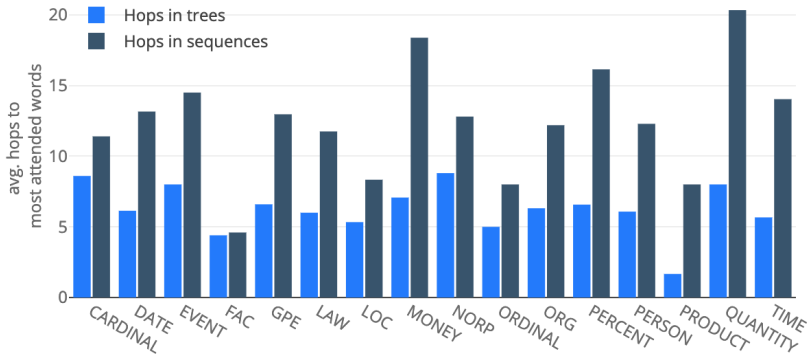
[7] Although in the example this is not problematic because the isolated roots do not carry context (i.e., it is mainly punctuation).

Table 4.3: The most attended words for nouns that are labeled as PERSON or LOCATION. The table shows the attended POS tag per entity, its coverage, and the corresponding list of words.

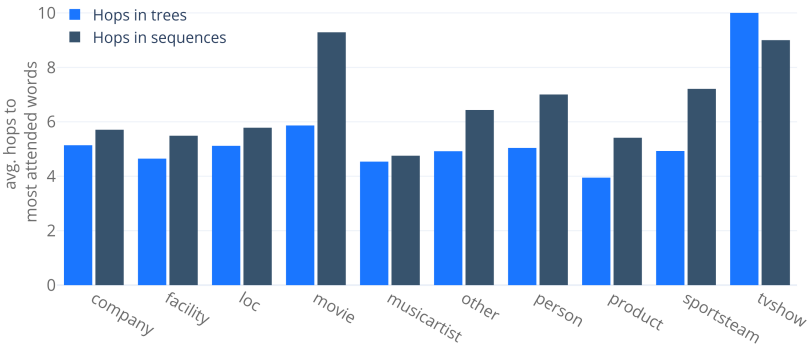
Entity	POS	Coverage	Most Attended Words
PERSON	NN	31.53%	*mr., correspondent, events, assistant, suit, case, ms., poll, a.m., consultant, president, express, officer, ignorance, cbs, wife, member, years, diet, menswear, life, relations, sunday, instance, season, network, school, woman, professor*
	VB	15.88%	*say, speak, come, go, write, begin, produce, need, tie, star, schedule, re-create, report, take, die, call, feature, love, continue, agree, know, pinch, consider, co-anchored, waive, brush, happen, issue, thank, welcome, slat, age, loot, resign, wrap, fall, chew, entomb, upset, allay, admire, place, strengthen*
LOCATION	VB	28.42%	*show, mellow, hold, direct, entitle, possess, continue, allow, travel, reach, begin, report, play, meet, carry, cause, miss, reduce, happen, mean, threaten, decide*
	NN	23.50%	*approaches, pollution, correspondent, temperatures, earthquake, u.s., meters, toepfer, run, re-enactment, behavior, mechanisms, breach, secrets, francisco, suspect, verdict, role, clue, analysis, thompson, space, story, zealand, violence, freedom, europe, front, glacier, conference, way, russia, death, cause, lake*

attended words and calculate the number of hops between the entities and the extracted words for both the sentence sequence and the dependency trees. Figure 4.6a clearly shows that the most attended words by the model are closer when their layout is a tree instead of a sequence. That partly explains a better performance of the TLSTM$_{\text{WORD+POS}}$ over the BLSTM$_{\text{WORD+POS}}$ for the SemEval-2010 dataset. This also aligns with the claim that the long-distance relations are potentially lost along the sentence when the words are treated as a straight sequence of tokens. However, the distances between the entity tokens and the most attended words is much less in the WNUT-2016 as shown in Figure 4.6b. This suggests that the BLSTM is hardly challenged for long-distant dependencies when processing the social media sequences. Hence, the gap in performance for the social media datasets is mainly due

to 1) the BLSTM being effective on short sequences, and 2) the TLSTM processing ill-formed dependency trees.



(a) SemEval-2010 dataset



(b) WNUT-2016 dataset

Figure 4.6: Average of hops to the most attended words in the validation set with respect to the entity tokens. For sequences, the hops are the number of tokens in between the entity and the most attended word. For trees, the hops are the number of nodes in the path that connects the entity and the most attended word.

Reliable dependency parses The TLSTM model exploits syntactic patterns, but the performance greatly relies on the quality of the dependency trees. I assess the impact of the dependency trees by evaluating the model using automatically generated trees from the Stanford CoreNLP tool [72]. I replace the trees in the validation set of the SemEval-2010 dataset, which originally has manually annotated dependency parses. Not surprisingly, the F_1 score drops by 3.31 absolute points from 86.49 to 83.18 F_1 score. By further inspection of the trees generated from scratch, I find that they contain multiple roots mainly because of

the multiple utterances that are transcribed from speech in the elaboration of these datasets. Multi-rooted trees affect the performance of the TLSTM model because they prevent connecting information across the sentence.

4.4 Limitations

The main limitation of this study lies in the quality of the dependency trees. The Tree-LSTM component works in a bottom-up fashion running from the leaves to the root of the tree. If the tree is multi-rooted (i.e., a sentence mistakenly decomposed in multiple trees), the sentence is segregated. That means that a tree's context never reaches the other trees' contexts, limiting the model's information compared to a sequential LSTM. The relative attention alleviates that, but the sequential LSTM performs better in such scenarios.

Another limitation of recursive modeling is parallelization. Parallelizing sequences in a batch of samples is trivial compared to parallelizing trees. For the experiments in this study, I parallelized trees by grouping samples based on the number of children at every node, prioritizing the nodes with the largest number. That means that traversing the tree in a bottom-up fashion becomes an iterative process. Nevertheless, the code adds a layer of complexity while still performing about 1.4 times slower than the sequential LSTM. This aspect can be a bottleneck in practice.

4.5 Conclusion

I proposed a novel approach that combines sequential and recursive linguistic properties for NER. The model uses syntactic and semantic features extracted from TLSTM and BLSTM, respectively. Then, I fed this information into the relative and global attention mechanisms. The relative attention allows the model to exploit linguistic properties over the sentence

with respect to every word, while the global attention combines those properties to balance semantic and syntactic patterns. By exploring the relationships among the entities and the most attended words, I found that the model learned to detect words that disclose information of the entity types based on their syntactic properties. Lastly, the performance of the TLSTM greatly relies on the quality of the dependency parses, which makes it hard to apply on social media data.

Chapter 5

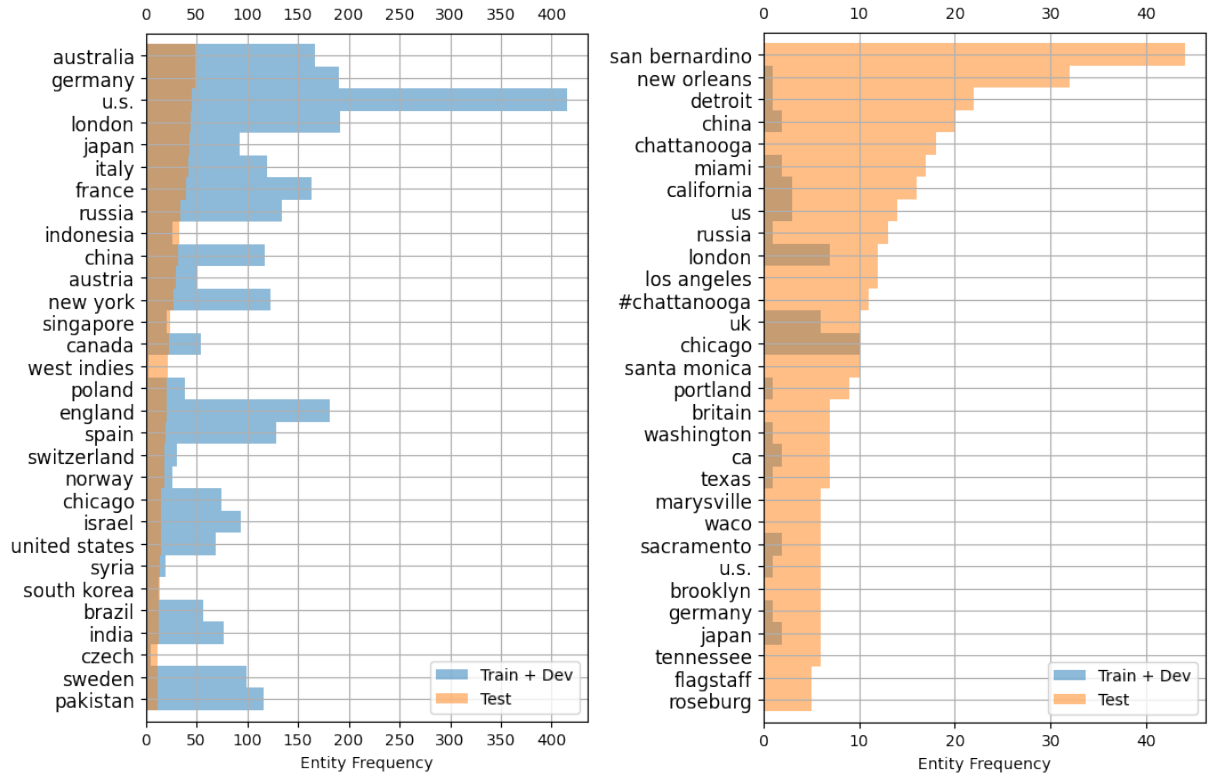
Are Models Relying on Memorization or Generalization?

Named entity recognition (NER) has seen significant performance improvements with the recent advances of pre-trained language models. Models such as ELMo [87], Flair [11, 10], and BERT [37] have pushed further the state of the art on the CoNLL-2003 [113] and WNUT-2016 [110] datasets. However, the large number of parameters of such models[1] raises the question on whether the boost in performance comes from **entity memorization** or **contextual generalization**. While both are important, the powerful memorization capabilities, stemming from millions of parameters, can misrepresent the model's genuine generalization proficiency.

The improvements in the CoNLL-2003 test set are widely considered as an indicator of progress in NER. Nevertheless, a further inspection of the test set using the subsets of **observed** (i.e., entities appearing in the training set) and **unobserved** entity instances (i.e., only appearing in the test set) is never conducted. This aspect is particularly relevant due

[1]ELMo, Flair LM, and BERT (large cased) have around 93.6M, 36.5M, and 333M parameters, respectively.

to the large overlap of entity instances among the partitions of the CoNLL-2003 dataset (see Figure 5.1), favoring memorization rather than generalization.



(a) Location class in CoNLL-2003.

(b) Geo-location class in WNUT-2016.

Figure 5.1: Top 30 most frequent entity instances in the (unobserved) test set and their overlap with the train and development sets (observed). The subfigure **(a)** shows the most common location entities from the CoNLL-2003 dataset, while the subfigure **(b)** describes de geo-location class in WNUT-2016. The x-axis denotes the frequency of the entity instances.

In contrast, the WNUT-2016 dataset is used as a benchmark less often than the CoNLL-2003 even though it poses more realistic challenges.[2] For instance, the WNUT-2016 dataset has a more diverse set of entity types (e.g., TV shows, movies, products, sports teams, etc.), extending the basic entity types in CoNLL-2003 (e.g., person, location, and organization). More importantly, the overlap of entity instances among partitions is extremely low compared

[2]One reason is that the CoNLL-2003 dataset focuses on the news domain, aiming at more general-purpose methods than the social media domain. Besides, the WNUT-2016 is very recent compared to the CoNLL-2003.

to the CoNLL-2003 dataset (see Figure 5.1).

In this study, I concentrate on three aspects to analyze whether the BERT model is generalizing or memorizing entity instances using the CoNLL-2003 and WNUT-2016 datasets. First, I analyze the datasets using statistics of the entity distributions (Section 5.1). The analysis corroborates whether the datasets promote memorization or generalization by evaluating the entity diversity among classes and the overlapping entity instances across partitions. Second, I fine-tune a pre-trained BERT model on each of the datasets. I investigate the performance of the fine-tuned models on observed and unobserved entity instances (Section 5.2). I derive insights from this exploration and propose methods to reduce the bias towards frequently observed entity instances (Section 5.3). The proposed methods show improvements for unobserved entity instances in the CoNLL-2003, but it is less effective in WNUT-2016 due to the low overlap of entities across partitions.

5.1 Data Analysis

I analyze the CoNLL-2003 and WNUT-2016 datasets to determine whether its evaluation favors memorization or generalization. I consider two sets within each dataset: train (train $\cup$ development) and test. I merge the train and development sets because they are usually overfitted when models are trained. The train set is directly used to update the model parameters (e.g., backpropagation [97]), while development is iteratively used to derive the best model. Also, the results on the test sets are ultimately what drive progress in these datasets.

Entity Overlap I define the entity instances appearing in the train set as the *observed entities*. The *unobserved entities* are the ones that belong to the test set and do not overlap with the observed set. Table 5.1 shows the overlap of repeated and unique entity instances

at the class level. The most overlapping entity types are location (85.9%) and geo-location (37.8%) for the CoNLL-2003 and the WNUT-2016 datasets. Note that the percentage for location is substantially greater than the one for geo-location, making the CoNLL-2003 class more prone to memorization. The CoNLL-2003 classes, except for the person class, exceed 50% of overlap between the observed and unobserved repeated entity instances. On the contrary, most of the WNUT-2016 classes are below 10% of overlapping entity instances.

Table 5.1: Statistics of the entity overlap with respect to the test set. **Overlap** describes the set operations: (Train ∪ Development) ∩ Test. **Repeated** means that the frequency of every entity is considered in the calculation, while **Unique** disregards the frequencies. Notably, the overlap percentages of the CoNLL-2003 are substantially higher than the ones from WNUT-2016. **Diversity** refers to the percentage of total unique entities out of the total repeated entities (e.g., 100% means all entities instances appear once). I sorted rows by overlap percentage from the repeated entities.

	Unique Entities		**Repeated Entities**		
Classes	**Total**	**Overlap (%)**	**Total**	**Overlap (%)**	**Diversity %**
CoNLL-2003					
Location	426	268 (62.9)	1,668	1,434 (85.9)	25.5
Other	274	116 (42.3)	702	483 (68.8)	39.0
Organization	759	409 (53.8)	1,617	971 (58.4)	46.9
Person	1,081	175 (16.1)	1,617	289 (17.8)	66.9
WNUT-2016					
Geo-location	432	85 (19.7)	882	333 (37.8)	49.0
Organization	306	18 (5.9)	621	138 (22.2)	49.3
Other	404	26 (6.4)	584	101 (17.3)	69.2
Person	416	28 (6.7)	482	48 (10.0)	86.3
Music artist	143	8 (5.6)	191	18 (9.4)	74.9
Sports team	128	6 (4.7)	147	11 (7.5)	87.1
Product	214	8 (3.7)	246	15 (6.1)	87.0
Facility	165	3 (1.8)	253	3 (1.2)	65.2
Movie	28	0 (0.0)	34	0 (0.0)	82.4
TV show	32	0 (0.0)	33	0 (0.0)	97.0

Entity Diversity I define class diversity as the percentage of unique entity instances among all (repeated) entities (see Table 5.1). The diversities for the most frequent classes

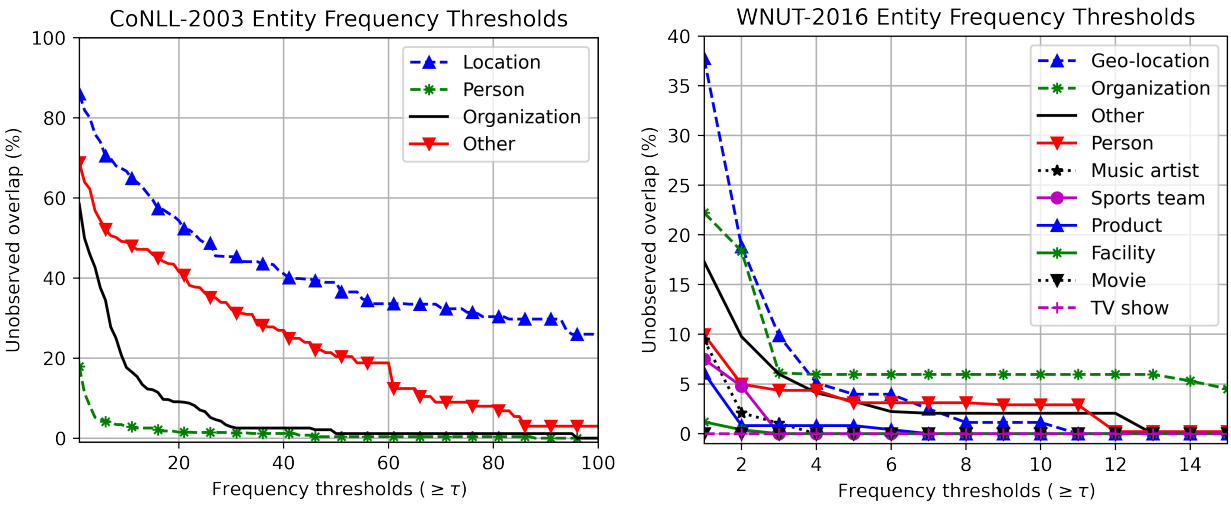
on each dataset are 25.5% for location (CoNLL-2003) and 49% for geo-location (WNUT-2016). Note that these classes are the least diverse in each dataset. Intuitively, suppose the diversity of a class is high. In that case, the evaluation of such class emphasizes the surrounding context more than the entity instance itself (i.e., less dependent on the actual entity instance), thus promoting generalization.

Entity Frequency The frequency of the overlapping entity instances is another factor that favors memorization. For instance, even if an observed entity happens in the test set, appearing hundred times in training is more prone to be memorized than occurring once. Considering that, Figure 5.2 shows the entity overlap between the train and test sets across different frequency thresholds (τ). For CoNLL-2003 (Figure 5.2a), note that the class person quickly decays by threshold $\tau = 5$ (e.g., entity instances from the test set that appear at least five times in the train set). On the other hand, the class location requires a higher threshold to decrease significantly (e.g., it does not reach 20% even with $\tau = 100$). For the WNUT-2016, most of the classes have reached 5% of overlap by $\tau = 4$.

5.2 Model Analysis: Fine-tuned BERT

In this section, I inspect the behavior of a fine-tuned BERT model on the CoNLL-2003 and WNUT-2016 datasets. I use BERT because it is a large model (333M parameters) capable of memorizing entity instances, and it generates contextualized word representations.

Evaluation by Subsets Table 5.2 shows the results of BERT on four evaluation sets. The first column describes the full test set that contains both observed and unobserved entities. For CoNLL-2003, the overall performance increases from 91.1% to 94.2% F_1 when the model is evaluated exclusively on observed entities (entity instances in the test set that appeared



(a) Overlap by thresholds for CoNLL-2003.

(b) Overlap by thresholds for WNUT-2016.

Figure 5.2: Overlap percentage of entity instances from the test set against the train and development sets. The x-axis denotes the frequency of the entity instances in the test. For instance, the frequency threshold $\tau = 1$ accounts for all the test entities that appear at least once in the train set.

at least once in the train set). However, the performance decays from 91.1% to 86.1% F_1 when the frequency of observed entities is limited up to five times. The model reaches the lowest performance when I evaluate it exclusively on unobserved entities, decaying from 91.1% to 84.5%. A similar pattern appears in the WNUT-2016, where the model reaches the best performance on the observed entities subset (from 50.8% to 71.2% F_1). However, the tendency compared to CoNLL-2003 changes in the observed entities whose frequencies are restricted to five times at most. The model performs better than in the full test set, improving from 50.8% to 69% F_1. Lastly, the unobserved entities show an overall decay from 50.8% to 44.8% F_1, but the class-level performance shows both increasing and decreasing scores. These results suggest that highly frequent entity instances are more impactful to the model (i.e., prone to memorization) than entity instances that appear a few times in the train set.

Although the observed and unobserved entity instances' overall tendencies are consistent

Table 5.2: Results on the full test set and its observed and unobserved entity subsets. For the observed subsets, ≥ 1 means that the entities appear in training at least once, whereas ≤ 5 refers to entities that appear in training at most five times. Note that there is a decreasing tendency (↓) in the scores of the unobserved entities with respect to the full dataset scores; the scores from the observed entities tend to increase (↑). The "N/A" entries for movie and TV show mean that the classes do not appear in such subsets.

Dataset	Classes	Test Set	Overlap. (≥ 1)	Overlap. (≤ 5)	Nonoverlap.	Diversity %
CoNLL-2003	Location	93.0	95.6 ↑	85.8 ↓	72.4 ↓	25.5
	Other	81.8	90.5 ↑	69.1 ↓	52.9 ↓	39.0
	Organization	88.4	93.9 ↑	87.6 ↓	84.3 ↓	46.9
	Person	95.9	96.5 ↑	95.0 ↓	95.2 ↓	66.9
	F_1 micro	**91.1**	**94.2** ↑	**86.1** ↓	**84.5** ↓	
	Support	5,648	1,685	524	973	
WNUT-2016	Geo-location	69.1	87.7 ↑	87.9 ↑	59.2 ↓	49.0
	Organization	57.8	82.0 ↑	72.1 ↑	45.1 ↓	49.3
	Other	32.4	34.4 ↑	38.6 ↑	31.6 ↓	69.2
	Person	63.9	69.1 ↑	66.7 ↑	67.1 ↑	86.3
	Music artist	18.4	33.3 ↑	25.0 ↑	16.0 ↓	74.9
	Sports team	48.2	57.1 ↑	66.7 ↑	48.1 ↓	87.1
	Product	14.1	61.5 ↑	54.5 ↑	7.7 ↓	87.0
	Facility	35.7	28.6 ↓	33.3 ↓	38.2 ↑	65.2
	Movie	6.0	N/A	N/A	3.4 ↓	82.4
	TV show	24.1	N/A	N/A	25.5 ↑	97.0
	F_1 micro	**50.8**	**71.2** ↑	**69.0** ↑	**44.8** ↓	
	Support	3,473	304	264	2,347	

with the entity frequencies, some classes are resilient to drop the performance. For instance, the person class slightly drops the performance from 95.5% to 95.2% in CoNLL-2003, while it increases the performance from 63.9% to 67.1% for WNUT-2016. Likewise, the sports team, facility, and TV show classes show this pattern. Not surprisingly, these classes are the ones with the highest rates of diversity, according to Table 5.1. This suggests that the evaluation for these entity classes balances fairly well the trade-off between memorization and generalization.

Evaluation by Frequency Thresholds Figure 5.3 shows the fine-tuned model over different subsets defined by a frequency threshold. For CoNLL-2003, the model stabilizes the performance after around threshold $\tau = 5$. Nevertheless, the model handles consistently well

the person class, keeping a high F_1 score from the beginning. These results reinforce the previous insights about the frequency impact in the model performance. For WNUT-2016, the frequencies do not impact the performance as much due to the less repeated entity instances.

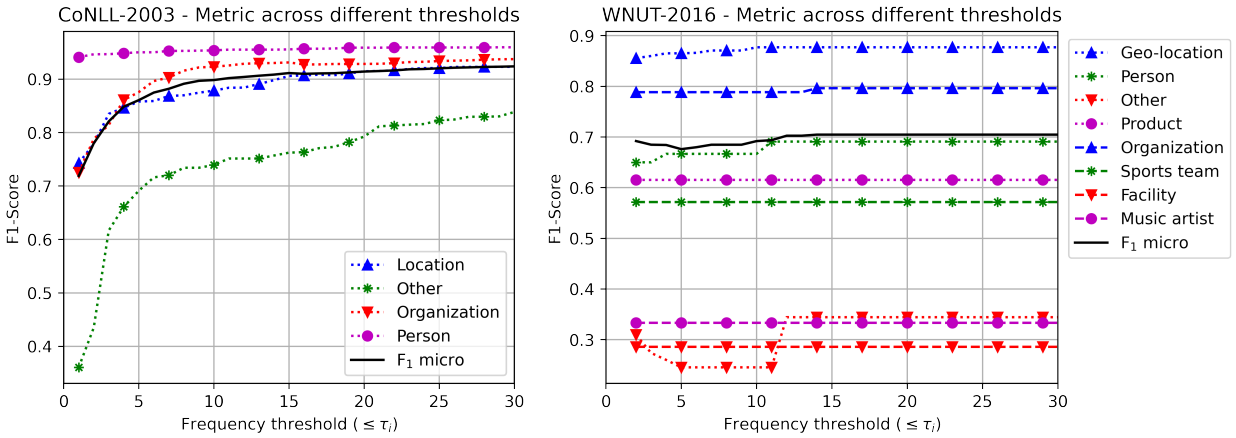


Figure 5.3: The BERT results on the observed subset for the CoNLL-2003 (left) and WNUT-2016 (right) datasets. The data points start at frequency one since this is the observed entity subset.

Context-dependent Entities Another interesting approach is to exchange entity instances of the same type between sentences. The idea is to use the test sentences containing observed entity instances (i.e., the **controlled sample**) to create a twin test set containing the unobserved entity instances instead (i.e., the **treatment sample**). This approach could show whether entities are context-dependent (i.e., generalized from context) or context-independent (i.e., memorized by the model) by performing similarly or differently on the controlled and treatment samples. However, swapping entity instances is a sensitive task because the text's overall meaning can change easily (e.g., the name of a restaurant vs. the name of a company when referred to its facilities). The scenario becomes more complicated when there are multiple entities in the same text, even if only one is changed (i.e., entity relations can be jeopardized). In addition, exchanging entities introduces alternative explanations (also known as covariates) to the treatment sample. I consider very simple

covariates, such as the length of the sentences, the entities' length, and the number of entity tokens in the sentence.[3]

In this experiment, the null hypothesis states that using the unobserved entities in the twin dataset does not explain the model's memorization or generalization capabilities. I create the twin dataset guiding the entity swaps by the cosine similarity between the observed entity and the pool of unobserved entities.[4] I consider very simple covariates that ended up validating the null hypothesis: sentence length, entity length, and the number of entity tokens within a sentence. To test whether these covariates are alternative explanations or not, I run a t-test comparing these statistics between the controlled and treatment samples. For both the CoNLL-2003 and the WNUT-2016 datasets, the covariates reached p-values < 0.001. That means that the differences between the controlled and the treatment samples are statistically significant between the covariates. Even though the F_1 performance of the model decayed when the entity instances were swapped (approximately 4% for CoNLL-2003, and 2% on WNUT-2016), it is not possible to conclude that the model is relying more on memorization than generalization.

5.3 Reducing the Memorization Bias

Considering the insights of the previous two sections, I explore five different approaches to balance the generalization-memorization trade-off.

Masked Entities (ME) I randomly mask entity instances at the word level (using the special token `[MASK]`) with probability 0.5 at every epoch during training. Even though the

[3]More covariates can be added, but these were sufficient to show that they can validate the null hypothesis.

[4]I tried multiple thresholds of cosine similarity where low scores allow for more candidates with less quality, but high scores easily reduce the pool to a few entities. Also, the cosine similarity is computed over the contextualized vector representations of the entities extracted from BERT.

input contains masked tokens, I still force the model to predict all the entity instances in the sentences, including the masked ones.

Frequency Loss Weights (FLW) I collect the frequency of every entity instance of the train set. Then, when I calculate the loss of the model at the token level, I weigh infrequent entity instances (≤ 3) with a penalty of $\lambda = 1.5$. Non-entities and frequent entity instances are not weighted ($\lambda = 1.0$).

Causal LM (CLM) Considering that BERT is a pre-trained LM, I define a secondary causal LM objective (i.e., the target is the input shifted one timestep) that preserves this knowledge during fine-tuning. This secondary loss prevents catastrophic forgetting [54] while also adapts to the data domain.[5] I weigh this loss with $\lambda =$ 1e-2 so that it does not interfere with the NER loss.

Parent LM (PLM) Given that dependency grammar can help on generalization [8, 123], I generate the dependency trees [34] of the train sentences and extract the parent connection of every entity token. Then, I mask the parent tokens with probability 0.25 and add a masked language model (MLM) objective [37] to predict such words.

Connection Distribution (CD) To explore further the previous idea, I consider the immediate dependency grammar connections of the entity tokens. For each entity token (query), the model generates a probability distribution over the sentence (context), similar to the attention mechanism [16, 69]. Then, I generate a uniform probability distribution over the immediate connections using the dependency grammar (with smoothing at 0.9). I optimize the model to decrease the KL-divergence between the uniform and model-generated

[5]The vocabulary is not learned from scratch since I initialize this output layer with BERT's word embedding layer.

distributions.

5.3.1 Results

Table 5.3 shows the results of the previous experimental settings. I highlight the best average F_1 score per column within the scope of each dataset. Also, note that I developed the approaches iterating over the development set until reaching the best scores according to the overall averaged F_1 score (i.e., without explicitly tracking the observed or unobserved F_1).[6] I also provide the results of the final models for each approach on the test set.

Table 5.3: Results of the proposed methods to reduce memorization. **Obs.** and **Unobs.** refer to the observed and unobserved entity subsets, and **All** means the full test set. I show the average of three runs with different random seeds and their std. deviation as a subscript.

		Test F_1		
Dataset	**Method**	**All**	**Overlapping**	**Nonoverlap.**
CoNLL-2003	Baseline	$\mathbf{91.4_{\pm 0.39}}$	$94.2_{\pm 0.05}$	$\mathbf{84.9_{\pm 0.96}}$
	Masked entities (ME)	$89.8_{\pm 0.35}$	$92.9_{\pm 0.08}$	$82.8_{\pm 1.95}$
	Freq. loss weights (FLW)*	$90.9_{\pm 0.55}$	$93.9_{\pm 0.43}$	$84.0_{\pm 2.07}$
	Parent LM (PLM)	$\mathbf{91.4_{\pm 0.35}}$	$\mathbf{94.6_{\pm 0.3}}$	$83.1_{\pm 0.66}$
	Conn. Distrib. (CN)	$90.8_{\pm 0.13}$	$93.8_{\pm 0.25}$	$83.3_{\pm 0.55}$
WNUT-2016	Baseline	$49.8_{\pm 1.07}$	$\mathbf{69.3}_{\pm 1.77}$	$43.9_{\pm 0.93}$
	Masked entities (ME)	$46.9_{\pm 1.21}$	$67.0_{\pm 1.59}$	$42.2_{\pm 0.48}$
	Freq. loss weights (FLW)*	$\mathbf{50.5_{\pm 1.04}}$	$67.0_{\pm 0.61}$	$\mathbf{45.4_{\pm 2.71}}$
	Parent LM (PLM)	$49.0_{\pm 2.15}$	$67.7_{\pm 3.36}$	$43.8_{\pm 1.71}$
	Conn. Distrib. (CN)	$48.4_{\pm 0.76}$	$67.6_{\pm 4.4}$	$43.1_{\pm 0.94}$

* Statistically significant with respect to the baseline with p-value < 0.01 in student's t-test.

The results for CoNLL-2003 show that there is no substantial difference among the approaches; the scores are very close to each other, and the standard deviations make the score ranges to overlap considerably. However, the WNUT-2016 results show different behavior.

[6] Also, note that the experimentation with the proposed methods uses the original splits in the datasets (i.e., the training set is no longer combined with the development set).

The baseline score from the development set (51.9%) is exceeded in many cases, being the FLW experiment with the best F_1 score (53.6%). One possible explanation for the disparate results between the datasets is the impact of pre-trained knowledge. That is, the methods have little or no effect on the CoNLL-2003 because BERT has been pre-trained extensively on very similar data (i.e., the same domain of the CoNLL-2003 dataset).[7] Thus, reducing the bias towards highly-frequent entity instances require an approach that reaches beyond the fine-tuning experiments I propose here. This explanation is far less applicable to the WNUT-2016 dataset because the social media domain deviates significantly from the news domain (e.g., misspellings and arbitrary abbreviations are present in named entities). Additionally, novel entities keep appearing on social media platforms (e.g., movie titles, TV shows, etc.).

The most effective methods for the unobserved entities are the FLW and CLM—they exceed the baseline scores significantly on both the development and test sets of the WNUT-2016 dataset. These methods, however, target fundamentally different aspects. The FLW experiment effectively reduces a tendency towards frequent entities by increasing the penalty of the token-level loss for the infrequent ones. On the other hand, the CLM experiment allows the model to transition towards the social media domain by multi-tasking between causal language modeling and the actual fine-tuning of the task.[8] Nevertheless, combining both experiments did not improve the performance for the WNUT-2016.

The ME and FLW methods target explicitly the problems I describe in Sections 5.1 and 5.2. However, the performance of the former shows less stable behavior than the latter; it only improves the development baseline for WNUT-2016 with a considerably large standard

[7]Note that BERT was pre-trained in 2018 using Wikipedia data, which extensively covers topics and entities included in the CoNLL dataset created in 2003.

[8]The causal language model objective ties the input subword embedding table in BERT with the parameters that project to the output vocabulary, so there is no need to learn those parameters from scratch.

deviation. This suggests that masking entity instances is a slightly drastic measure. Reasonably, the FLW approach has a more effective and stable performance because it prioritizes more frequent entity instances without totally ignoring highly frequent entities.

I believe that an extension of the test dataset that includes more unobserved entities could help determine these methods' effectiveness since the current subsets are small.

The least effective method is the PLM experiment. The PLM method is designed to exploit the entity's parent words as they appear in dependency trees. Intuitively, BERT can predict such words because it was pre-trained with a masked language model objective. However, the results do not support that intuition in either of the datasets. A masked LM objective over contextual words (i.e., the parent words) is counterproductive in this case—the model tries to predict contextual words that are masked while simultaneously exploiting the remaining context to learn the named entity task.

The CN method aims at a similar approach but without masking the context. This method is less drastic than the PLM experiment. It emphasizes the immediate connections (i.e., the entity's parent and children nodes) while still having the full context in the sentence for the entity task. The CN result (52.2%) for WNUT-2016 is better than the PLM (50.4%) and baseline (51.9%) results. This tendency is also consistent in the unobserved category between the CN and PLM experiments (49.8% vs. 46.8%). Nevertheless, the test set does not show the same behavior as the development set.

5.4 Analysis

Class-level insights The results of the WNUT-2016 unobserved test set showed a sensible gap with respect to the FLW results (43.9% vs. 45.4%). Table 5.4 provides class-level details of the F_1 scores between the baseline and the FLW experiments.[9] Notably, the gap between

[9]I take the best performer among the three runs per method.

the two methods remains consistent among most of the classes. For example, the results for sports team, person, facility, and product improve over the baseline while also among the most diverse classes with many instances. The most challenging class is movie since it is the most diverse type in the set. However, the number of movie instances is too low to draw a conclusion.

Table 5.4: Class-level F_1 scores for the baseline and the frequency loss weight (FLW) experiments on the unobserved test set. I also provide the number of instances and the diversity percentage per class. The rows are sorted descendingly by the number of instances. Note that the diversity rate is the lowest for the company class, while the scores per model are almost the same.

	F_1 Measure				
Class	**Baseline**	**FLW**	**Difference**	**Instances**	**Diversity %**
Geo-location	59.2	59.1	-0.1	439	49.0
Other	31.6	33.3	+1.7	381	69.2
Organization	45.1	50.1	+5.0	378	49.3
Person	67.1	69.7	+2.6	375	86.3
Facility	38.2	45.7	+7.5	218	65.2
Product	7.7	11.4	+3.7	216	87.0
Music artist	16.0	25.9	+9.9	148	74.9
Sports team	48.1	53.7	+5.6	129	87.1
TV show	25.5	12.2	-13.3	33	97.0
Movie	3.4	3.8	+0.4	30	82.4
Overall	**44.8**	**47.4**	**+2.6**	**2,347**	N/A

Error analysis Although the FLW experiment improves upon the baseline, there are many mistakes made by the model. The FLW model makes more mistakes by confusing the entity classes instead of inadvertently missing entities compared to the baseline (i.e., the FLW model has better recall). A total of 1,276 tokens were incorrectly classified as non-entities (i.e., false negative) by the FLW model. At the same time, the baseline predicted 1,470 false-negative samples. Additionally, the baseline model predicts correctly 21,101 non-entity tokens while the FLW model gets right 21,042. While these numbers are ultimately about

mistakes, the results show that the FLW model is trying to pick up more entity signals from the text than the baseline.

The tendency of picking up more entity instances than the baseline is also captured in the examples of Table 5.5. For example, even though the fourth sentence only has music artists according to the ground-truth, the FLW model wrongly detects Bibby and Fab as the person class. The baseline predictions, on the other hand, do not detect any of those two entity instances. In any case, the FLW mistakes are understandable: the model does not find enough information in the context to conclude that the person entities are music artists. Conversely, the last entity, Drake, is even in the model's vocabulary, meaning that the model saw this entity during pre-training (i.e., this entity appeared in Wikipedia data used to pre-trained BERT). This fact allows both models to predict the entity Drake as a music artist correctly. Also, this explains why the model behaves differently from the previous entity instances compared to the last one. Bibby and Fab are not in the vocabulary of the model, and the model proceeds to split the words into subwords so that they are representable.

Table 5.5: The table shows cases where the FLW model gets the predictions wrong. The sentences are tagged with their corresponding ground-truth entities in brackets (e.g., [Penders Field]$_{\text{facility}}$). Also, I provide the results of the baseline for comparison.

#	Sentence	FLW Entities	Baseline Entities
1	Graduation has been set!! The **[SHS]$_{\text{facility}}$** CLASS OF 2015 will graduate on June 18th, 6:00 p.m. at **[Penders Field]$_{\text{facility}}$**!	[SHS CLASS]$_{\text{other}}$ [Penders Field]$_{\text{facility}}$	[Penders Field]$_{\text{facility}}$
2	Do you rmbr i spoke on **[Maggi]$_{\text{company}}$** and suspicious emptying of market fr a new player perhaps??!	[Maggi]$_{\text{person}}$	–
3	**[Jean Geoffroy joue Bach]$_{\text{person}}$** (CD, May-2013, 2 Discs, **[Skarbo]$_{\text{organization}}$**)	[Jean Geoffroy]$_{\text{person}}$ [Bach]$_{\text{product}}$ [Skarbo]$_{\text{music-artist}}$	[Jean Geoffroy]$_{\text{person}}$ [Skarbo]$_{\text{organization}}$
4	**[Bibby]$_{\text{music-artist}}$** dropped, **[Fab]$_{\text{music-artist}}$** dropped, **[Drakes]$_{\text{music-artist}}$** album coming soon but where **[frank ocean]$_{\text{music-artist}}$** at? We're about 5 months into June	[Bibby]$_{\text{person}}$ [Fab]$_{\text{person}}$ [Drakes]$_{\text{music-artist}}$	[Drakes]$_{\text{music-artist}}$
5	On another hand, **[Mayim]$_{\text{person}}$** will be guesting on **[Access Hollywood Live]$_{\text{tvshow}}$** tomorrow !	[Mayim]$_{\text{person}}$ [Access Hollywood Live]$_{\text{tvshow}}$	[Mayim]$_{\text{person}}$ [Access Hollywood]$_{\text{other}}$

Besides ambiguity among entity instances, the predictions also show errors related to the BIO (Beginning, Inside, Outside) scheme (e.g., an inside label is not consistent with

the previous entity class in the sequence). This is a persistent problem among the models, regardless of the methods proposed in this study. In any case, this can be easily solved by applying conditional random fields (CRF) on the output layer, as shown in previous chapters. Nevertheless, I leave this component out to isolate BERT without any additional parameter as well as to preserve the flexibility of studying different objective functions.[10]

Sentence length Long sentences are more likely to provide more context. This suggests that models that discourage frequent entity instances and rely on context for their predictions should reach better performance on longer sentences. To measure that, I split the unobserved test set into quartiles according to the distribution of the sentence lengths (i.e., the first quartile is up to 13 tokens; the rest are up to 18, 22, 35 tokens). Surprisingly, the performance did not show a clear tendency concerning the length. That is, the baseline and the FLW models show the same F_1 score tendency across quartiles. The FLW F_1 scores are 48%, 44%, 47% and 50% while the baseline shows 43%, 41%, 45%, and 47%.

After manually inspecting the predictions, I noticed that the length is not a good indicator of context in this data in many cases. Many of the tokens are often noise or do not incorporate context to the sentences (e.g., interjections, punctuation, emojis, etc.) while inflating the sentences' length. In any case, there are a few words that can still convey enough information to infer entity classes (e.g., in Table 5.5, the model detects that the word Maggi is a person using contextual words such as spoke).

[10]The CRF negative log-likelihood loss is computed using the Viterbi algorithm and dynamic programming. This makes the alteration of the normalization factor impractical, so an approximation of this factor is required [33].

5.5 Limitations

One of the limitations of this study is that it only evaluates the memorization scenario in the model's fine-tuning stage. While there is evidence that highly frequent entity instances of the CoNLL-2003 and WNUT-2016 appear in pre-training (e.g., entity instances are in the vocabulary words), it is not clear how much they impact the memorization or generalization capabilities of the model. Nevertheless, that requires more investigation, and it goes beyond the scope of the study from this chapter.

Regarding unexplored hyper-parameters, it is important to note that the hyper-parameter decisions were made based on the data's insights and model behavior. The experimentation took placed using a group of candidate hyper-parameters rather than single numbers. Nevertheless, more aspects need to be explored, such as the frequency factor λ used in the FLW experiments. For example, dynamic weights could have been applied when multiple losses were involved. That said, the results shown in this study could have a scope for improvement by just tuning hyper-parameters.

5.6 Conclusions and Future Work

I studied two of the primary named entity recognition datasets from the news and social media domains: CoNLL-2003 and WNUT-2016. I exposed the memorization-generalization trade-offs in the datasets, and I showed evidence that the CoNLL-2003 has a sensible bias towards observed entity instances (i.e., entity instances that appear during training). In addition to the datasets, I studied the behavior of a fine-tuned BERT model. Lastly, I proposed simple methods to alleviate the bias towards memorization. The methods showed more effective on the WNUT-2016 dataset and did not make significant changes in the CoNLL-2003 dataset.

Part II

Sequence Labeling on Linguistic Code-Switching

Chapter 6

LinCE: A Centralized Linguistic Code-Switching Evaluation Benchmark

Many researchers have proposed novel methods to handle code-switched data, showing improvements on core NLP tasks such as language identification (LID), named entity recognition (NER), and part-of-speech (POS) tagging. However, many of these approaches are usually evaluated on a few language pairs and a specific domain, and it is not clear whether these models are exclusive to such scenarios or they can generalize to other tasks, domains, and language pairs. Furthermore, choosing the best-published model for benchmarking purposes is not an easy task either. These problems exist mainly because 1) there is no official benchmark for general code-switching evaluation that allows direct comparisons across multiple tasks, and 2) methods are usually not comprehensively evaluated across datasets with different language pairs.

I propose a centralized **Lin**guistic **C**ode-switching **E**valuation (**LinCE**) benchmark to

overcome these problems. I have consolidated a benchmark from preexisting corpora considering the following aspects: 1) multiple language pairs from high- and low-resource languages, 2) typologically-diverse languages,[1] 3) various NLP tasks, and 4) different social media domains. LinCE is comprised of four LID datasets, two POS tagging datasets, three NER datasets, and one sentiment analysis (SA) dataset, providing a total of ten datasets (see Table 6.1). Furthermore, an important contribution of LinCE is the new stratification process to provide fair and, in some cases, official splits for the tasks at hand. This required a careful inspection of the original datasets from which I list five major issues (see Section 6.2.5) and propose new splits for nine out of the ten datasets. Moreover, LinCE is publicly available at `ritual.uh.edu/lince` as this benchmark continues to grow and include new tasks and language pairs in the future.

Table 6.1: Overview of the LinCE language pairs and tasks.

Language Pair	**LID**	**POS**	**NER**	**SA**
Spanish-English	✓	✓	✓	✓
Hindi-English	✓	✓	✓	-
Nepali-English	✓	-	-	-
MS Arabic-Egyptian Arabic	✓	-	✓	-

6.1 Linguistic Challenges

Although code-switching can happen in more than two languages, this benchmark focuses on language pairs only. The frequent alternations between two languages is precisely what makes the automated processing of code-switching data difficult. I quantify such complexity using the CMI index proposed by [44] as shown in Table 6.2. The higher the CMI index, the more alternations the dataset contains, and hence, the more complex the code-switching

[1] I also consider the geolocation of such languages to account for places across the world.

Table 6.2: The CMI scores and the number of tokens across corpora. **All Posts** describes the number of posts in the corpora and **All CMI** is the corresponding CMI scores for such samples. Similarly, **CS Posts** denotes the number of code-switched posts (excluding monolingual posts) and **CS CMI** is the corresponding CMI scores for such samples. I also show the number of tokens that belong to the language pairs (**Lang1**, **Lang2**) as well as the overall number of tokens (**All Tokens**), which includes other LID labels beyond the language pairs. English is the **Lang1** class for English-paired languages; for MSA-EA, Modern Standard Arabic is the **Lang1** class. I omit the CMI information for the MSA-EA NER corpus because the corpus does not come with language identification labels.

Task	Corpus	Languages	All Posts	All CMI	CS Posts	CS CMI	Lang1	Lang2	All Tokens
LID	[78]	SPA-ENG	32,651	8.29	12,380	21.86	129,065	170,793	390,953
	[106]	NEP-ENG	13,011	19.85	10,029	25.75	59,037	78,360	188,784
	[73]	HIN-ENG	7,421	10.14	3,317	22.68	84,752	29,958	146,722
	[78]	MSA-EA	11,243	2.82	1,326	23.89	140,057	40,759	227,354
POS	[104]	HIN-ENG	1,489	20.28	1,077	28.04	12,589	9,882	33,010
	[108]	SPA-ENG	42,911	24.19	41,856	24.81	178,135	92,517	333,069
NER	[1]	SPA-ENG	67,223	5.49	17,466	21.16	163,824	402,923	808,663
	[103]	HIN-ENG	2,079	19.99	1,644	25.28	13,860	11,391	35,374
	[1]	MSA-EA	12,335	–	–	–	–	–	248,478
SA	[83]	SPA-ENG	18,789	20.70	18,196	21.37	65,968	144,533	286,810

behavior is. In addition to the alternation of languages, I briefly describe other linguistic challenges that each specific language pair poses to current NLP systems:

- **Spanish-English (SPA-ENG)**. While English is a Germanic language, a significant number of words from its current vocabulary have been borrowed from Latin and French since the Middle Ages [114]. This particular set of words tends to overlap with words from Spanish, a Latin-based language. This overlap increases ambiguity and directly affects systems that rely on character-based approaches, for example, in the case of language identification. Code-switching also appears within the words, often inflecting words by conjugating English verbs using Spanish grammatical rules. This behavior is known as Spanglish [95], and it particularly affects non-contextualized word embeddings as it increases the out-of-vocabulary (OOV) rate.

- **Hindi-English (HIN-ENG)**. One of the most challenging aspects of this language

pair is the lack of a standardized transliteration system. Speakers transliterate Hindi employing mostly ad-hoc phonological rules to use the English alphabet when writing. Using the same Roman alphabet makes code-switching more convenient but the lack of an official standard for transliteration makes it difficult to process with existing resources exclusively available for Hindi with the Devanagari script. Furthermore, although Hindi loosely follows the subject-object-verb (SOV) structure, its flexible word order poses an additional challenge to NLP systems.

- **Nepali-English (NEP-ENG)**. Similar to HIN-ENG, Nepali is transliterated using the English alphabet when code-switched with English. This behavior makes Nepali speakers to write driven by arbitrary phonological rules that allow the romanization of Nepali using the English alphabet, which excludes the few monolingual resources available for Nepali. Also, Nepali is a subject-object-verb (SOV) language while English is subject-verb-object (SVO). This grammatical difference intuitively encourages more code-switching points since it is proven that, when code-switching occurs, the languages involved still preserve their grammatical structure [107], which forces more fine-grained alternations to obey the SOV and SVO structures. In practice, I notice a large code-switching rate for Nepali-English captured by the averaged CMI index in Table 6.2, being one of the largest scores while having a corpus of middle size.

- **Modern Standard Arabic-Egyptian Arabic (MSA-EA)**. Arabic is well known for its diglossia [40], which combines a number of Arabic dialects with Modern Standard Arabic within the same community. This combination of dialects enables a large occurrence of linguistic code-switching. One of the main challenges with this language pair is that there is a significantly large word overlap while the word meanings can vary depending on the language. Even more, Arabic is a morphologically rich language and

it allows multiple word orders, which increases the semantic complexity for NLP systems. Additionally similar to Spanglish, code-switching can occur at the morpheme level, where speakers often add morphological inflections to nouns.

6.2 Tasks

LinCE is built upon four tasks and four language pairs to provide a total of eleven datasets. In Sections 6.2.1 to 6.2.4, I discuss the datasets used for every task. Then, in Section 6.2.5, I describe and justify the modifications to nine out of the ten datasets in order to establish official splits that can be adopted for this benchmark. Lastly, in Section 6.2.6, I explain the evaluation criteria to rank the leaderboard in the LinCE platform.

6.2.1 Language Identification (LID)

Handling code-switched data requires to identify the languages involved. The task of language identification (LID) is one of the first steps that validates whether a system can handle code-switched data or not. Correctly classifying the language associated to text units (e.g., words or sub-word tokens) enables to process code-switched text in higher-level applications where general language understanding takes place. LinCE uses preexisting datasets for the language identification task. Specifically, in this version of LinCE, I focus on the language pairs Spanish-English, Hindi-English, Nepali-English, and Modern Standard Arabic-Egyptian Arabic. I briefly explain each corpus below, and for some of them, I propose new splits as explained in the stratification section (Section 6.2.5). Figure 6.1 shows the final distribution of the labels across the LID corpora used in LinCE. Also, these datasets follow the CALCS LID label scheme, which is `lang1`, `lang2`, `mixed` (partially in both languages), `ambiguous` (either one or the other language), `fw` (a language different than `lang1` and

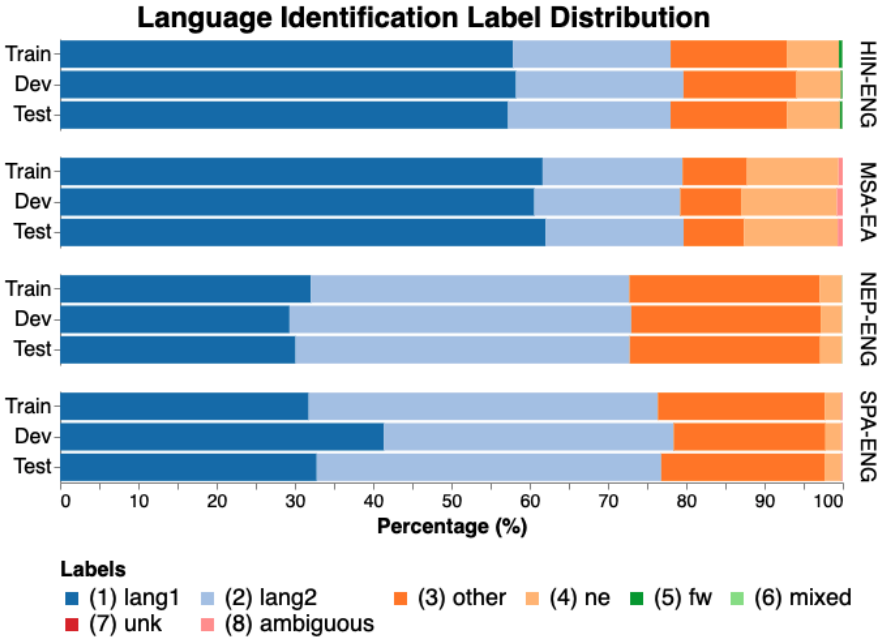


Figure 6.1: LID label distribution used in LinCE. While HIN-ENG and SPA-ENG have very few tokens for `unk` and `fw` (<1%), MSA-EA and NEP-ENG do not have occurrences of such labels. Also, with the exception of MSA-EA, all the partitions are proposed for LinCE as described in Section 6.2.5.

`lang2`), `ne` (named entities), `other`, and `unk` (unrecognizable words).

- **SPA-ENG**. I use the Spanish-English corpus from the 2016 CALCS workshop [78]. This corpus uses Twitter data and it contains 32,651 posts that are comprised of 390,953 tokens. I provide new splits for this corpus because the original splits do not have a similar label distribution and the label `fw` does not appear in the development set.

- **HIN-ENG**. I use the Hindi-English corpus released by [73]. This corpus uses Twitter and Facebook data, which have been partly collected and partly re-used from the ICON 2016 competition [101]. The corpus contains a total 7,421 posts comprised of 146,722 tokens. Also, I proceed with the stratification process on this corpus because the length of the posts were not considered while doing the splits; Twitter has a character

length limit in its post, whereas Facebook posts do not have such restriction resulting in significantly longer text. Moreover, the labels `ambiguous` and `unk` do not appear in the development set.

- **NEP-ENG**. The Nepali-English corpus comes from the 2014 CALCS workshop [106]. This corpus was collected from Twitter and it contains 13,011 posts and 188,784 tokens. I perform a stratification process to provide standard splits for this corpus since the organizers only provided train and test, and the test set does not include any occurrence of the `ambiguous` class.

- **MSA-EA**. I use the Modern Standard Arabic-Egyptian Arabic corpus from the 2016 CALCS workshop [78]. This corpus contains Twitter data and it is comprised of 11,243 tweets with 227,354 tokens. Note that there is no occurrence of the labels `fw` and `unk` in the entire corpus. I propose new partitions due to the variation across distributions for both the LID labels as well as sentence lengths.

6.2.2 Parts-of-Speech (POS) Tagging

Part-of-speech (POS) tagging is an important linguistic component that enables more sophisticated syntactic analysis such as constituency and dependency parsing. Code-switched data is not exempted of such analysis. In fact, previous studies have shown that syntax is preserved and compliant with the syntactic rules of the individual languages when code-switching occurs [107]. In this benchmark, I consider the language pairs Hindi-English and Spanish-English (see proposed label distribution in Figure 6.2):

- **HIN-ENG**. [104] provides 1,489 tweets (33,010 tokens) annotated with POS tags and three language IDs (`hi` for Hindi, `en` for English, and `rest` for any other token). The POS tags are annotated using the universal POS tagset proposed by [89] with the

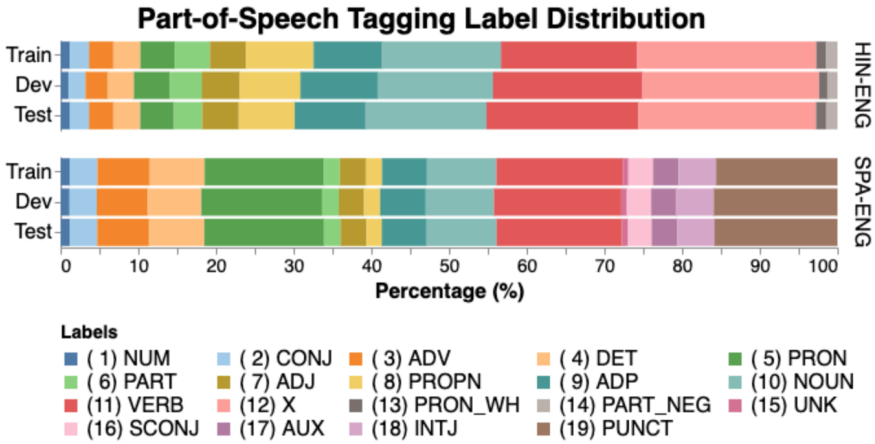


Figure 6.2: POS label distribution used in LinCE. The partitions for both datasets are proposed for LinCE as described in Section 6.2.5. Note that the labels `UNK`, `SCONJ`, `AUX`, `INTJ`, and `PUNCT` only appear in the SPA-ENG corpus, whereas `PRON_WH` is unique for HIN-ENG.

addition of two labels: `PART_NEG` and `PRON_WH`. The corpus does not provide training, development, and test splits due to the small number of samples. However, for the purposes of the benchmark, I propose standard splits using the stratification criteria discussed in Section 6.2.5.

- **SPA-ENG**. I use the Miami Bangor corpus with the annotations provided by [108]. The Bangor corpus is composed of bilingual conversations from four speakers with a total of 42,911 utterances and 333,069 tokens. The corpus contains POS tags from the universal POS tagset and LID labels. The LID labels are `eng` for English, `spa` for Spanish, `eng&spa` for mixed or ambiguous words, and `UNK` for everything else. Additionally, I proceed with the stratification process to provide the official training, development, and testing sets for this benchmark since the original sets were split by speakers.

6.2.3 Named Entity Recognition (NER)

Named entity recognition (NER) is another important core NLP task that enables higher-level applications such as question-answering, semantic role labeling, and information extraction. LinCE covers NER for three languages pairs: Spanish-English, Modern Standard Arabic-Egyptian Arabic, and Hindi-English (see proposed label distribution in Figure 6.3):

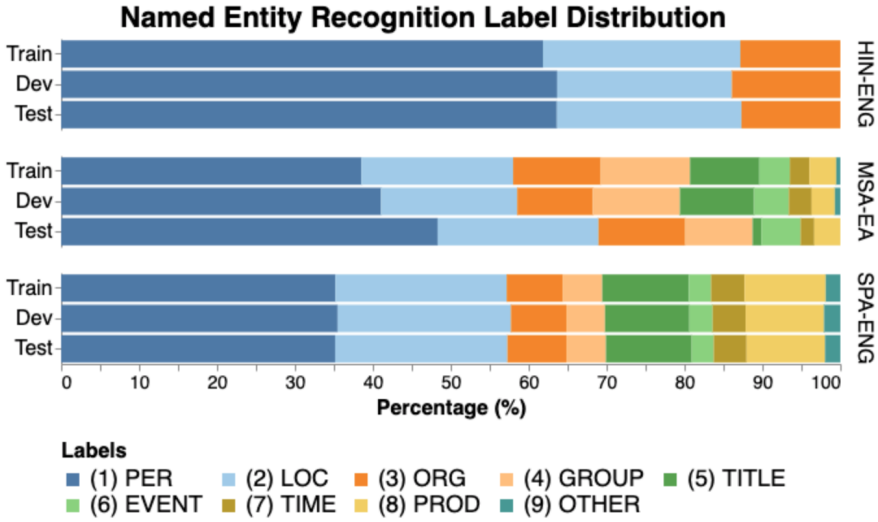


Figure 6.3: NER label distribution used in LinCE. All the datasets have the BIO scheme, but I only show the entity types for simplicity. Note that HIN-ENG only contains `PER`, `LOC`, and `ORG`. Also, with the exception of MSA-EA, all the other partitions are proposed for LinCE as described in Section 6.2.5.

- **SPA-ENG**. This corpus was introduced in the 2018 CALCS competition for NER [1], and it contains a total of 67,223 tweets with 808,663 tokens. The labels are `organization`, `person`, `location`, `group`, `product`, `title`, `event`, `time`, and `other`. Along with the NER labels, I have added the LID categories for every token, which follows the CALCS LID scheme. Moreover, I propose new splits for this corpus since the distribution of the NER labels across the splits is not consistent to the one from the full corpus. Additionally, the original development set is significantly small compared to the other splits, only accounting for 832 tweets, and the LID labels were not

taken into consideration for the splitting process (e.g., the label `fw` does not appear in the development set). I provide new splits following the stratified process described in Section 6.2.5

- **MSA-EA**. This corpus was also introduced in the 2018 CALCS competition for NER, following the same entity label scheme as in the SPA-ENG corpus. The corpus uses the tweets from the 2016 CALCS LID dataset to form the training and development sets. While the LID labels are available for the training and development splits, the test set was annotated only using the NER labels. Thus, this is the only corpus for which I do not consider the language identification analysis. The corpus contains 12,335 posts and 248,452 tokens. I adopt the splits provided by the organizers during the 2018 CALCS competition.

- **HIN-ENG**. This corpus is proposed by [103], and it is composed of 2,079 tweets with 35,374 tokens. The dataset has been annotated with both NER and LID labels. The entity labels are `person`, `location`, and `organization`, while the LID labels are `eng` (English), `hin` (Hindi), and `rest` (any other token). This dataset is small, and for that reason, the authors opted to do 5-fold cross validation instead of partitioning the dataset. Nevertheless, for the sake of the benchmark, I split the data using the stratification process that fairly splits the dataset accounting for LID and NER label distributions, as well as the distribution of the tweet lengths.

6.2.4 Sentiment Analysis (SA)

I choose sentiment analysis as the fourth benchmark task to incorporate a high-level NLP application in contrast to the previous core NLP tasks. I use the Spanish-English corpus provided in the SentiMix competition [83]. The organizers reduce the monolingual posts,

increasing the number code-switched instances. Table 6.2 shows that this language pair is the second highest scores on the "All CMI" column for the sentiment analysis task. The task requires to predict one of the sentiments `positive`, `negative`, or `neutral` for every post. Additionally, this corpus is annotated with LID labels at the token level, following the CALCS LID scheme, and it contains 18,789 tweets comprised of 286,810 tokens. I propose new partitions for this dataset to correct the label distribution from the original splits (see Figure 6.4).

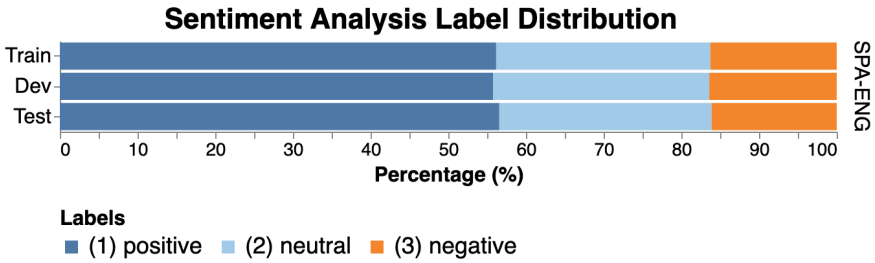


Figure 6.4: Label distribution of the sentiment analysis corpus used in LinCE. Note that this distribution differs from the origial dataset.

6.2.5 Stratification

For nine out of ten datasets,[2] I propose new splits that lead to a more appropriate evaluation (see Table 6.3 for a high-level distribution). I provide new splits for datasets where I found at least one of the following issues:

1. At least one of the splits does not have one or more classes. That is, one or more classes from are not evaluated at all in the development or test set.

2. The distribution of the label set for a given task is substantially different across splits or against the full corpus distribution (i.e., when merging all the splits into a single set).

[2]I did not partition the NER MSA-EA dataset because it does not have LID labels, which is essential to keep the code-switching behavior balanced across splits in the stratification process.

Table 6.3: Final data distribution of the LinCE benchmark. Note that the proposed distribution follows the stratification process described in Section 6.2.5, which generates partitions that differ from the original datasets.

			Training			Development			Test		
Tasks	**Corpus**	**Languages**	**CMI**	**Posts**	**Tokens**	**CMI**	**Posts**	**Tokens**	**CMI**	**Posts**	**Tokens**
LID	[78]	SPA-ENG	8.491	21,030	253,221	7.062	3,332	40,391	8.264	8,289	97,341
	[106]	NEP-ENG	20.322	8,451	122,952	17.079	1,332	19,273	19.754	3,228	46,559
	[73]	HIN-ENG	10.222	4,823	95,224	10.122	744	15,446	9.930	1,854	36,052
	[78]	MSA-EA	2.567	8,464	171,872	3.185	1,116	21,978	3.849	1,663	33,504
POS	[104]	HIN-ENG	21.449	1,030	22,993	15.293	160	3,476	18.910	299	6,541
	[108]	SPA-ENG	24.191	27,893	217,068	24.040	4,298	33,345	24.282	10,720	82,656
NER	[1]	SPA-ENG	5.567	33,611	404,428	4.398	10,085	122,656	5.867	23,527	281,579
	[103]	HIN-ENG	20.117	1,243	21,065	19.913	314	5,364	19.733	522	8,945
	[1]	MSA-EA	–	10,103	204,296	–	1,122	22,742	–	1,110	21,414

3. The length of the sentences do not follow a similar distribution across splits or against the full corpus. This is a relevant criteria to consider since length is positively correlated with context, and less or more context can make a huge different for tasks such as NER.

4. The NER, POS, and SA datasets contain LID labels that were not considered during the time of the stratification process, potentially affecting the balance of code-switching occurrences across the splits.

5. There is no official split for the training, development and test sets to provide the scope of fair comparison.

For the datasets where I find at least one of these issues (see Table 6.4), I proceed to stratify based on the language identification labels, if available, the task-specific labels (i.e., for tasks other than LID), and the lengths of the sentences. Note that providing splits that consider these three factors jointly in the stratification process is not trivial. In fact, in the case of sequence labeling tasks, I may have multiple non-unique labels per sentence, which constraints the ability to draw a distribution similar to the full corpora (e.g., adding a sentence impacts the distribution of different labels occurring in the same sentence).

Table 6.4: The table shows the datasets for which I propose new splits. The column **Reason** provides the reason number according to the aspects listed in Section 6.2.5. The lower the KL-divergence, the more similar the splits are to the full corpus distribution.

			KL-divergence	
Task	**Dataset**	**Reason**	**before**	**after**
LID	SPA-ENG	1, 2	0.10586	0.00528
	HIN-ENG	1, 2, 3	4.64265	0.00064
	NEP-ENG	1, 3, 5	0.00552	0.00059
	MSA-EA		0.17737	0.00026
POS	SPA-ENG	4, 5	0.00140	0.00005
	HIN-ENG	5	–	0.00133
NER	SPA-ENG	1, 2, 4	0.00239	0.00001
	HIN-ENG	5	–	0.00007
SA	SPA-ENG	2, 4	0.09579	0.00002

To provide splits considering these three criteria, I follow the iterative stratification process proposed by [99]. This process targets multi-label data, which is a different scenario for the document and sequence labeling classification datasets used in LinCE. To adapt the sequence labels to the multi-label scenario, I treat a post as a single sample that is associated to a group of labels. In the case of tasks other than LID, I gather the LID labels with the task-specific labels (i.e., NER, POS, or SA labels) into a single group of unique labels. I also incorporate sentence lengths to the label set of a sample by choosing one of three length categories: `small` ($\leq$ 10 tokens), `medium` (>10 and $\leq$ 20 tokens), or `large` (>20 tokens). For instance, the SPA-ENG NER sample

"LREC$^{event}_{ne}$ será$_{lang2}$ hosted in Marseille$^{location}_{ne}$"

English: *"LREC will be hosted in Marseille"*

has the set of unique labels {`lang1`, `lang2`, `ne`, `event`, `location`, `O`, `small`}, where the first three labels are for LID, the following three labels are for NER, and the last one represents the sentence length (note that the order and the repetitions of the labels do not matter).

Once I have the set of labels associated to a single sample (e.g., a set of LID, POS, and length labels), I can follow the iterative stratification processed used for multi-label classification on the corpus. I have found that this procedure works well in practice; I measure the KL-divergence of the label distributions from each of the splits against the distribution of the full corpus before and after the stratification, and I found that the proposed splits have less divergence (see Table 6.4). While KL-divergence is not often employed to corroborate the distributions of a stratified corpus, I use the divergence score to quantify whether the distribution of the full corpus has been preserved in the proposed splits, and whether the new splits are better distributions than the original splits. The final numbers of sentences and tokens per partition are listed in Table 6.3.

6.2.6 Evaluation

LinCE adopts an evaluation model similar to SemEval, Kaggle, and GLUE [118]. The LinCE platform is hosted at `ritual.uh.edu/lince` where participants are able to upload their predictions for the test data on each task. The platform will score the submissions and publish the results in a public leaderboard for each task. The leaderboard is ranked by the average of the task scores. Table 6.5 shows the leaderboard as of October 22nd, 2020.

Table 6.5: The LinCE leaderboard as of October 19th, 2020. The top three entries in the table are the baseline models using off-the-shelf models publicly available. The char2subword mBERT model is a sequence labeling method that I proposed in Chapter 8. The missing scores denote that the participants did not provide results on those task. Hence, the average score is also skipped.

		LID				**POS**		**NER**			**SA**
Model	**Avg**	**SPA**	**HIN**	**NEP**	**MSA**	**SPA**	**HIN**	**SPA**	**HIN**	**MSA**	**SPA**
mBERT	82.23	**98.36**	94.24	96.32	**91.55**	**97.07**	86.30	64.05	72.57	65.39	56.43
BERT	81.70	98.35	**96.40**	**96.46**	88.36	96.92	87.02	61.15	**74.46**	59.44	**58.40**
ELMo	78.98	97.93	95.43	95.90	86.53	96.34	86.71	52.58	68.79	56.68	52.88
Char2Subword mBERT	-	98.33	96.23	96.19	91.19	96.88	**88.23**	**64.65**	73.38	**66.13**	-
Spanish BERT	-	-	-	-	-	-	-	-	-	-	56.47

6.3 Limitations

The main limitation of this work is the empirical approach to determine a good data split. I compare the KL-divergence between the resulting data splits before and after the proposed stratification method, noting a large difference in the distribution in most of the datasets. However, I do not specify a KL-divergence threshold to determine when a split is optimal. This leaves room for improvements on new approaches that reach optimal splits.

6.4 Conclusion

I introduced the **Lin**guistic **C**ode-switching **E**valuation (**LinCE**) benchmark using ten publicly available datasets. In addition, I reviewed such datasets and found important issues that undermined the evaluation process (e.g., labels not appearing in the test set, or substantially different distributions among splits, etc.). Then, I proposed new splits using a novel stratification technique with up to three criteria (e.g., LID labels, task-specific labels, and sentence lengths). I showed the distribution of the full corpus is preserved in the proposed splits used in LinCE, which is not the case in most of the original partitions. Finally, I expect that LinCE will be well-received by the NLP community, and the platform will keep evolving with the incorporation of more tasks and language pairs in the near future.

Chapter 7

From English to Code-Switching

I study the CS phenomenon using English as a starting language to adapt the models to multiple code-switched languages, such as Nepali-English, Hindi-English and Spanish-English. In the first part of this chapter, I focus on the task of language identification (LID) using ELMo [87] as the reference for English knowledge. The hypothesis is that English pre-trained models should be able to recognize whether a word belongs to English or not when such models are fine-tuned with code-switched text (see examples in Figure 7.1).

Spanish-English Tweet
@USER$_{\texttt{other}}$ @USER$_{\texttt{other}}$ *go too* cavenders$_{\texttt{ne}}$ y tambien ve a @USER$_{\texttt{ne}}$ 👌$_{\texttt{other}}$
English: @USER @USER go to cavenders and also go to @USER 👌

Hindi-English Tweet
Keep calm and keep kaam se kaam !!!$_{\texttt{other}}$ *#office #tgif #nametag* #buddha$_{\texttt{ne}}$ *#SouvenirFromManali #keepcalm*
English: Keep calm and mind your own business !!!

Nepali-English Tweet
Youtube$_{\texttt{ne}}$ ma live re ,$_{\texttt{other}}$ chalcha ki vanni aash garam !$_{\texttt{other}}$ *Optimistic* .$_{\texttt{other}}$
English: They said Youtube live, let's hope it works! Optimistic.

Figure 7.1: Examples of code-switched tweets and their translations from the CS LID corpora for Hindi-English, Nepali-English and Spanish-English. The subscript `ne` refers to named entities and `other` is used for punctuation, emojis or usernames. English text appears in *italics* and other languages are underlined.

To accomplish that, I introduce CS-ELMo, an extended version of ELMo that contains a position-aware hierarchical attention mechanism over ELMo's character n-gram representations. These enhanced representations allow the model to see the location where particular n-grams occur within a word (e.g., affixes or lemmas) and to associate such behaviors with one language or another.

In the second part, I demonstrate the effectiveness of the CS-ELMo models by further fine-tuning them on tasks such as NER and POS tagging. Specifically, I show that the resulting models significantly outperform multilingual BERT and their homologous ELMo models directly trained for NER and POS tagging.

7.1 Methodology

ELMo is a character-based language model trained on a large amount of English data that provides deep contextualized word representations [87]. ELMo forms its word vectors by extracting morphological information out of characters, which is essential for this study since certain character n-grams can reveal whether a word belongs to one language or another. In the following sections I discuss the position-aware hierarchical attention mechanism that allows it to adapt to code-switching settings.

7.1.1 Position-aware Hierarchical Attention

ELMo convolves character embeddings in its first layers and uses the resulting convolutions to represent words. While this process has proven effective in practice, it has the following shortcomings:

1. Convolutional networks do not account for the positions of the character n-grams (i.e., they do not preserve the sequential order), losing linguistic properties such as affixes.

2. ELMo down-samples the outputs of its convolutional layers by max-pooling over the feature maps. However, this operation is not ideal to adapt to new morphological patterns from other languages as the model tends to discard patterns from languages other than English.

To address these aspects, I introduce CS-ELMo, an extension of ELMo that incorporates a position-aware hierarchical attention mechanism that enhances ELMo's character n-gram representations. This mechanism is composed of three elements: position embeddings, position-aware attention, and hierarchical attention.

Position embeddings. Consider the word $\mathbf{x}$ of character length l, whose character n-gram vectors are $(x_1, x_2, \ldots, x_{l-j+1})$ for an n-gram order $j \in \{1, 2, \ldots, n\}$.[1] The n-gram vector $x_i \in \mathbb{R}^c$ is the output of a character convolutional layer, where c is the number of output channels for that layer. Also, consider n position embedding matrices, one per n-gram order, $\{\mathbf{E}_1, \mathbf{E}_2, \ldots, \mathbf{E}_n\}$ defined as $\mathbf{E}_j \in \mathbb{R}^{(k-j+1)\times e}$ where k is the maximum length of characters in a word (note that $l \leq k$), e is the dimension of the embeddings and j is the specific n-gram order. Then, the position vectors for the sequence $\mathbf{x}$ are defined by $\mathbf{p} = (p_1, p_2, \ldots, p_{l-j+1})$ where $p_i \in \mathbb{R}^e$ is the i-th vector from the position embedding matrix $\mathbf{E}_j$. I use $e = c$ to facilitate the addition of the position embeddings and the n-gram vectors.[2]

Position-aware attention. Instead of down-sampling with the max-pooling operation, I use an attention mechanism similar to the one introduced by [16]. The idea is to concentrate mass probability over the feature maps that capture the most relevant n-gram information along the word, while also considering positional information. At every individual n-gram

[1] ELMo has 7 character convolutional layers, each layer with a kernel size from 1 to 7 characters ($n = 7$).
[2] ELMo varies the output channels per convolutional layer, so the dimensionality of $\mathbf{E}_j$ varies as well.

order, the attention mechanism uses the following equations:

$$u_i = v^\intercal \tanh(\mathrm{W}_x x_i + p_i + b_x) \tag{7.1}$$

$$\alpha_i = \frac{\exp(u_i)}{\sum_{j=1}^{N} \exp(u_j)}, \quad \text{s.t.} \sum_{i=1} \alpha_i = 1 \tag{7.2}$$

$$z = \sum_{i=1} \alpha_i x_i \tag{7.3}$$

where $W_x \in \mathbb{R}^{a \times c}$ is a projection matrix, a is the dimension of the attention space, c is the number of channels for the specific n-gram order, and p_i is the position embedding associated to the n-gram x_i. $v \in \mathbb{R}^a$ is the attention vector to be learned, and α_i is a scalar that describes the attention probability associated to the i-th n-gram. z is the weighted sum of the input character n-gram vectors and the attention probabilities, which is the down-sampled word representation for the n-gram order n. Note that this mechanism is used independently for every order of n-grams resulting in a set of m vectors $\{z_1, z_2, \ldots, z_m\}$ from Equation 7.3. This allows the model to capture relevant information across individual n-grams before they are combined (i.e., all bi-grams, all tri-grams, etc.).

Hierarchical attention. The previous mechanisms handle the problems aforementioned. That is, I have considered positional information as well as the attention mechanism to down-sample the dimensionality. These components retrieve one vector representation per n-gram order per word. While ELMo simply concatenates the n-gram vectors of a word, I experiment with another layer of attention that can prioritize n-gram vectors across all the orders. I use Equation 7.1 without p_i, and in Equation 7.3, instead of doing the weighted sum, I concatenate the weighted inputs. This concatenation keeps the original dimensionality expected in the upper layers of ELMo, while it also emphasizes which n-gram order should receive more attention.

7.1.2 Sequence Tagging

Following [87] on sequence labeling, I use the ELMo with a bidirectional LSTM layer and a conditional random field (CRF) on top. This is the baseline architecture and I build upon it. Figure 7.2 shows the overall CS-ELMo model architecture (Figure 7.2A) and the details of the proposed component inside it (Figure 7.2B).

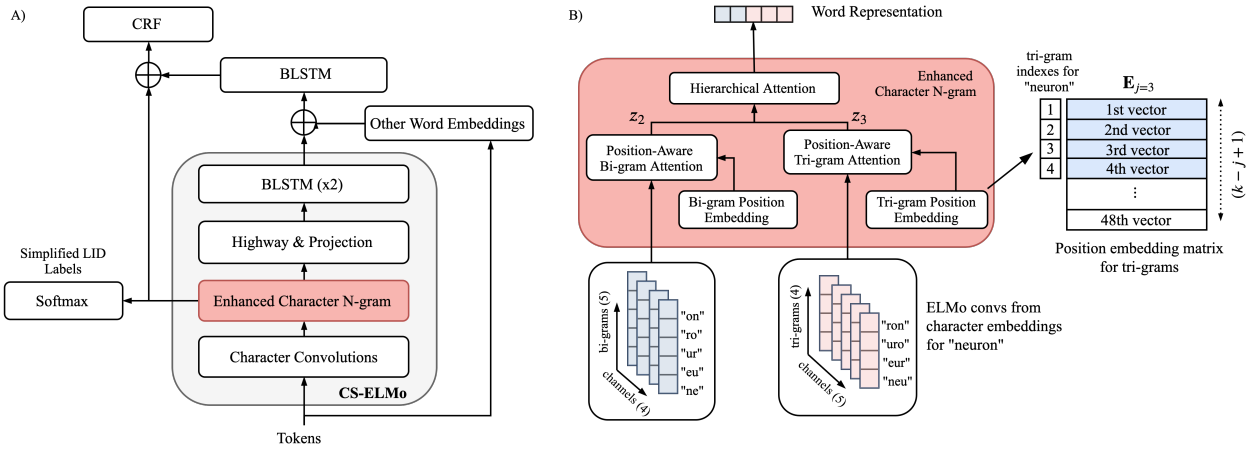


Figure 7.2: A) The left figure shows the overall model architecture, which contains ELMo followed by BLSTM and CRF, and a secondary task with a softmax layer using a *simplified* LID label set. The largest box describes the components of ELMo, which include the enhanced character n-gram module proposed in this paper. B) The right figure describes in detail the enhanced character n-gram mechanism inside ELMo. The figure shows the ELMo convolutions of a word as input and a single vector representation as output.

The first modification is the concatenation of static English word embeddings to ELMo's word representation, such as Twitter [86] and *fastText* [21] embeddings similar to [54] and [73]. The second modification is the concatenation of the enhanced character n-gram representation with the input to the CRF layer. This emphasizes even further the extracted morphological patterns, so that they are present during inference time for the task at hand (i.e., not only LID, but also NER and POS tagging). The last modification is the addition of a secondary task on a *simplified*[3] language identification label scheme, which only uses the

[3]The full LID label set uses eight labels (`lang1`, `lang2`, `mixed`, `ambiguous`, `fw`, `ne`, `other`, and `unk`), but I only consider three labels (`lang1`, `lang2` and `other`) to predict LID based on characters.

output of the enhanced character n-gram mechanism. Intuitively, this explicitly forces the model to associate morphological patterns (e.g., affixes, lemmas, etc.) to one or the other language.

Multi-task learning. I train the model by minimizing the negative log-likelihood loss of the CRF classifier. Additionally, I force the model to minimize a secondary loss over the *simplified* LID label set by only using the morphological features from the enhanced character n-gram mechanism (see the softmax layer in Figure 7.2A). The overall loss $\mathcal{L}$ of the model is defined as follows:

$$\mathcal{L}_{task_t} = -\frac{1}{N}\sum_{i}^{N} y_i log(\hat{y}_i) \tag{7.4}$$

$$\mathcal{L} = \mathcal{L}_{task_1} + \beta\mathcal{L}_{task_2} + \lambda\sum_{k} w_k^2 \tag{7.5}$$

where $\mathcal{L}_{task_1}$ and $\mathcal{L}_{task_2}$ are the negative log-likelihood losses defined by Equation 7.4. $\mathcal{L}_{task_1}$ is the loss of the primary task (i.e., LID, NER, or POS tagging), whereas $\mathcal{L}_{task_2}$ is the loss for the *simplified* LID task weighted by β to smooth its impact on the model performance. The third term accounts for ℓ_2 regularization, and λ is the penalty weight.

7.2 LID Experiments

Approach 1. I establish three strong baselines using a vanilla ELMo (experiment 1.1), ELMo combined with BLSTM and CRF (experiment 1.2), and multilingual BERT (experiment 1.3) [37].

Approach 2. In the second set of experiments, I add the components of the mechanism upon ELMo combined with BLSTM and CRF (experiment 1.2). I start by replacing the max-pooling operation with the attention layer at every individual n-gram order in experiment 2.1. In experiment 2.2, I incorporate the position information. Experiment 2.3 adds the hierarchical attention across all n-gram order vectors. I apply the mechanism for n-gram orders in the set {1, 2, 3}, which I report in Table 7.1.

Table 7.1: The results of incremental experiments on each LID dataset. The scores are calculated using the weighted F-1 metric across the eight LID labels from CALCS. Within each column, the best score in each block is in **bold**, and the best score for the whole column is underlined. Note that development scores from subsequent experiments (e.g., experiments 2.2 and 2.3) are statistically significant with p-value < 0.02.

Exp. ID	Experiment	Nepali-English		Spanish-English		Hindi-English	
		Dev	Test	Dev	Test	Dev	Test
Approach 1 *(Baseline models)*							
1.1	ELMo	96.192	95.700	95.508	96.363	95.997	96.420
1.2	ELMo + BLSTM + CRF	**96.320**	95.882	95.615	**96.748**	**96.545**	**96.717**
1.3	ML-BERT	95.436	**96.571**	**96.212**	96.212	95.924	96.440
Approach 2 *(Upon experiment 1.2)*							
2.1	Attention on each n-gram	96.413	96.771	95.952	96.519	96.579	96.069
2.2	Position-aware attention on each n-gram	96.540	96.640	95.994	**96.791**	96.629	96.141
2.3	Position-aware hierarchical attention	**96.582**	**96.798**	**96.072**	96.692	**96.705**	**96.186**
Approach 3 *(Upon experiment 2.3)*							
3.1	Concatenating character n-grams at the top	96.485	96.761	96.033	96.775	96.665	96.188
3.2	Adding simplified LID (secondary) task	96.612	96.734	96.051	96.932	96.565	96.215
3.3	Adding static word embeddings	**96.879**	**97.026**	**96.757**	**97.532**	**96.776**	**97.001**
Comparison: *Previous best published results*							
Conditional random fields [73]		-	-	96.510	97.060	96.6045	96.840

Approach 3. The third set of experiments emphasizes the morphological clues extracted by the previous mechanism (experiment 2.3). First, in experiment 3.1, I concatenate the enhanced character n-grams with their corresponding word representation before feeding the input to the CRF layer. In experiment 3.2, I add the secondary task over the previous experiment to force the model to predict the simplified LID labels by only using the morphological clues (i.e., no context is provided). Finally, in experiment 3.3, I add static word embeddings

that help the model to handle social media style and domain-specific words.

The best results are delivered by experiment 3.3, which outperforms both the baselines and the previous state of the art on the full LID label scheme (see Table 7.1).

7.3 POS Tagging and NER Experiments

I use LID to adapt the English pre-trained knowledge of ELMo to the code-switching setting. Once this is achieved, I fine-tune the model on downstream NLP tasks such as POS tagging and NER. In this section, the goal is to validate whether the CS-aware ELMo model can improve over vanilla ELMo, multilingual BERT, and the previous state of the art for both tasks. More specifically, I use the best architecture (experiment 3.3) from the LID experiments 1) without the code-switching adaptation, 2) with the code-switching adaptation and only retraining the inference layer, and 3) with the code-switching adaptation and retraining the entire model.

POS tagging experiments. Table 7.2 shows the experiments on POS tagging using the Hindi-English dataset. When I compare the CS-ELMO + BLSTM + CRF model without CS adaptation (experiment 4.1) against the baseline (ELMo + BLSTM + CRF), the performance remains similar. This suggests that the enhanced n-gram mechanism can be added to ELMo without impacting the performance even if the model has not been adapted to CS. Slightly better performance is achieved when the CS-ELMo has been adapted to code-switching, and only the BLSTM and CRF layers are retrained (experiment 4.2). This result shows the convenience of the model since small improvements can be achieved faster by leveraging the already-learned CS knowledge while avoiding to retrain the entire model. Nevertheless, the best performance is achieved by the adapted CS-ELMO + BLSTM + CRF when retraining the entire model (experiment 4.3). The results are better than the baselines

Table 7.2: The F_1 scores on POS tagging for the Hindi-English dataset. CS knowledge means that the CS-ELMo architecture has been adapted to code-switching by using the LID task.

POS System	**Development F_1**	**Test F_1**
ML-BERT	86.84	84.70
ELMo + BLSTM + CRF	87.42	88.12
Prev. SOTA [104]	-	90.20
Architecture: CS-ELMo + BLSTM + CRF		
Experiment 4.1: No CS knowledge	87.02	87.96
Experiment 4.2: CS knowledge frozen	89.55	89.92
Experiment 4.3: CS knowledge trainable	**90.37***	**91.03***

* Statistically significant with respect to the ELMo + BLSTM + CRF baseline, with p-value < 0.01 [38]

and the previous state of the art.

Interestingly, the model improves over multilingual BERT, which is a powerful and significantly bigger model in terms of parameters. The intuition is that this is partly due to the word-piece tokenization process combined with the transliteration of Hindi. The fact that I use the multilingual version of BERT does not necessarily help to handle transliterated Hindi, since Hindi is only present in BERT's vocabulary with the Devanagari script. In contrast, ELMo generates contextualized word representations out of characters, which makes the model more suitable to adapt to the transliteration of Hindi.

NER experiments Table 7.3 shows the experiments on NER using the 2018 CALCS Spanish-English dataset. Similar to experiment 4.1, experiment 5.1 shows a performance close to the ELMo baseline without the CS adaptation. In contrast to experiment 4.2, experiment 5.2, which uses CS knowledge and only retrains the inference layer, slightly drops the performance of the model. The intuition is that, unlike POS tagging, the LID task is not inherently tied to the NER task. Despite of that, when I use CS knowledge and retrain the full model (experiment 5.3), I see a significant improvement of about 5% absolute

Table 7.3: The F_1 scores on the Spanish-English NER dataset. CS knowledge means that the CS-ELMo architecture has been adapted to code-switching by using the LID task.

NER System	**Development F_1**	**Test F_1**
ML-BERT	61.11	64.56
ELMo + BLSTM + CRF	59.91	63.53
Best at CALCS [115]	-	63.76
Prev. SOTA [121]	-	66.63
Architecture: CS-ELMo + BLSTM + CRF		
Experiment 5.1: No CS knowledge	62.59	66.30
Experiment 5.2: CS knowledge frozen	**64.39***	**67.96***
Experiment 5.3: CS knowledge trainable	64.28	66.84

* Statistically significant with respect to the ELMo + BLSTM + CRF baseline, with p-value < 0.0025 [38]

points on the F_1 metric over the model without CS adaptation (experiment 5.1). The only difference between experiment 5.1 and 5.3 is the presence of CS knowledge (i.e., ELMo is fine-tuned on LID), which emphasizes the importance of the transfer learning step to the CS setting.

7.4 Analysis

Position embeddings Localizing n-grams within a word is an important contribution of the method. I explore this mechanism by using the CS fine-tuned ELMo to predict the *simplified* LID labels on the validation set from the secondary task (i.e., the predictions solely rely on morphology) in two scenarios. The first one uses the position embeddings corresponding to the actual place of the character n-gram, whereas the second one chooses position embeddings randomly. I notice a consistent decay in performance across the language pairs, and a variation in the confidence of the predicted classes. The most affected language pair is Spanish-English, with an average difference of 0.18 based on the class probability gaps between both scenarios. In contrast, the probability gaps in Hindi-English and Nepali-English

are substantially smaller; their average differences are 0.11 and 0.09, respectively.

Attention analysis Figure 7.3 shows the tri-gram attention weights in the Spanish-English LID dataset. The model is able to pick up affixes that belong to one or the other language. For instance, the tri-gram *-ing* is commonly found in English at the end of verbs in present progressive, like in the word *com**ing*** from the figure, but it also appears in Spanish at different places (e.g., ***ing**eniero*) making the position information relevant. On the contrary, the tri-grams *aha* and *hah* from the figure do not seem to rely on position information because the attention distribution varies along the words.

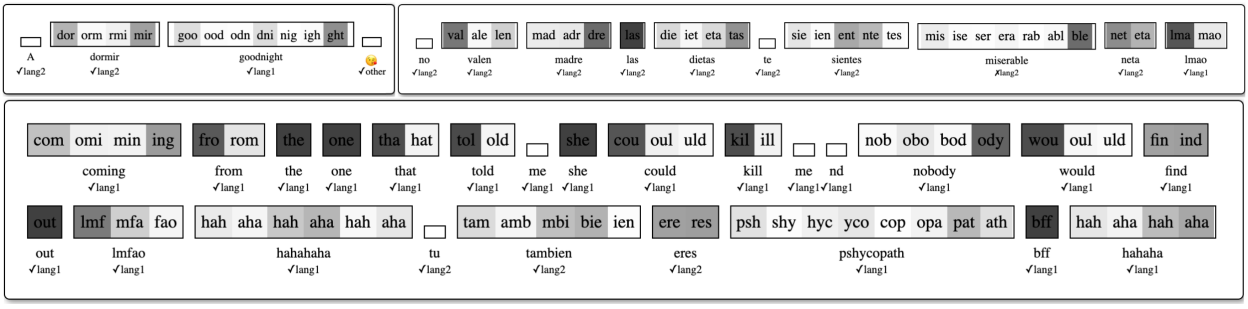


Figure 7.3: Visualization of the tri-gram attention weights for the 2016 Spanish-English LID dataset. The boxes contain the tri-grams of the word below them along with the right (✓) or wrong (✗) predictions by the model.

Error analysis Morphology is very useful for LID, but it is not enough when words have similar spellings between the languages. I inspect the predictions of the model, and find cases where, for example, *miserable* is labeled as `ambiguous` but the model predicts a language (see the top-right tweet in Figure 7.3). Although I find similar cases for Nepali-English and Hindi-English, it mostly happens for words with few characters (e.g., *me*, *to*, *use*). The model often gets such cases mislabeled due to the common spellings in both languages. Although this should be handled by context, the contribution relies more on morphology than contextualization, which I leave for future work.

7.5 Limitations

The main limitation of this study is that the model is only applicable to English-paired code-switched languages. That is because the pre-trained knowledge of the model only relies on English text. In the proposed method, English is an essential part of performing code-switching adaptation. Nevertheless, code-switching happens in many more language pairs that exclude English, limiting the proposed approach's scope.

Additionally, the adaptation of the model relies on labeled data. This means that the method not only requires English-paired code-switched text, but it also requires annotated data for language identification. If the ultimate goal is to target downstream NLP tasks such as NER or POS tagging, then it would be costly to annotate language identification and the task of interest. Despite that, the proposed method serves as evidence that it is possible to leverage pre-trained knowledge and adapt it to code-switching settings effectively.

7.6 Conclusion

I presented a transfer learning method from English to code-switched languages using the LID task. The method enables large pre-trained models, such as ELMo, to be adapted to code-switching settings while taking advantage of the pre-trained knowledge. I established new state of the art on LID for Nepali-English, Spanish-English, and Hindi-English. Additionally, I showed the effectiveness of the CS-ELMo model by further fine-tuning it for NER and POS tagging. The model outperforms multilingual BERT and homologous ELMo models on Spanish-English NER and Hindi-Enlgish POS tagging.

Chapter 8

From Multilingualism to Code-Switching

Considering that linguistic code-switching is inherently multilingual, it is natural to think of multilingual models as strong approaches to this phenomenon. While this is a valid assumption, there are a few caveats to consider. For instance, code-switching frequently appears as transliterated text when the scripts of the code-switched languages differ. That is, code-switchers conveniently rely on practical linguistic aspects to import a language into a non-native script (see Figure 8.1).[1] Such transliterated text prevents multilingual models from leveraging their knowledge initially captured in the native scripts.

Figure 8.1: A Hindi-English code-switched tweet. Hindi is transliterated to the Roman script (highlighted) instead of using Devanagari. Trans.: *Keep calm and mind your own business!!!*

[1] E.g., Hindi can be transliterated from Devanagari to the Roman script using phonological writing.

Another important aspect is the input representation of multilingual models. Recent methods, such as multilingual BERT (mBERT) [37] and XLM-RoBERTa [67], rely on the byte-pair encoding (BPE) algorithm [100] and a vocabulary of subword pieces to represent the input text. Although these methods alleviate the out-of-vocabulary (OOV) problem, they are susceptible to character-level modifications. If the model's vocabulary does not contain a given word, it gets split into pieces until the BPE algorithm finds matches of subwords within the vocabulary.[2] This process can break down words into sub-pieces that do not necessarily resemble morphological inflections (see Figure 8.2). Such a problem is significantly aggravated in the social media domain since the noise from user-generated texts (e.g., arbitrary spellings, abbreviations, and slang) can drastically affect the resulting sequence of subwords.

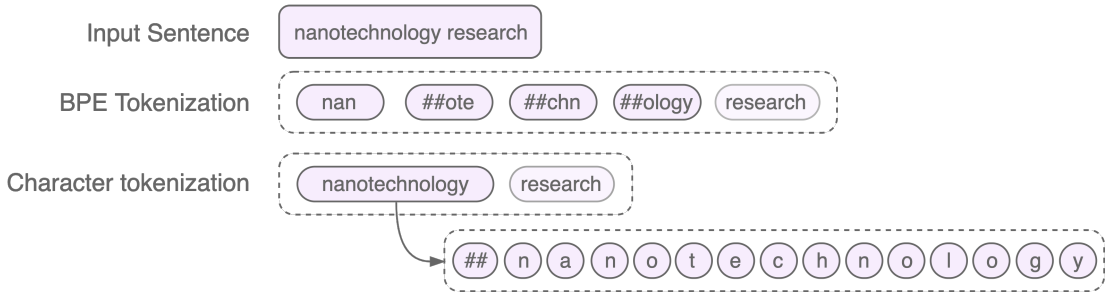


Figure 8.2: A tokenization example that compares BPE and character tokenization given an input sentence. For BPE, only the word *nanotechnology* is split because it does not appear in the vocabulary. This word is broken down into pieces that only resemble the morpheme *-ology* (i.e., *nano-tech-nology*). Also, note that the resulting subwords do not use the word *technology* as a piece despite being present in the vocabulary. For characters, the tokenization process does not segment words into subword pieces; instead, it provides a list of character sequences where each sequence represents a word.

Motivated by those challenges, I proposed a character-to-subword (char2subword) module that extends the embedding space of the subword lookup table in mBERT. Although the mBERT subword embedding space is continuous, the vector representations used by the model are restricted only to the entries in its vocabulary. This restriction no longer applies

[2]BPE greedily looks for matches of subword pieces bottoming down to characters.

when using the char2subword module; the char2subword module allows mBERT to extend its original vocabulary to flexible word representations never projected in the input embedding space before. The module takes the sequence of characters from a subword and produces its corresponding embedding vector, effectively resembling the original subword embedding table in mBERT (Section 8.1.1). This module becomes robust to spelling alterations by introducing character-level noise during training in the embedding approximation phase. Additionally, I integrate this module with the transformer layers of mBERT even further by resuming pre-training (Section 8.1.2). Once the char2subword module is adapted to the pre-trained language model, I evaluate the overall model performance by fine-tuning it on downstream tasks (Section 8.1.3). I show the method's effectiveness by outperforming the original mBERT model in the linguistic code-switching evaluation (LinCE) benchmark.

8.1 Method

Given the word w, a subword model produces a sequence of word pieces $\boldsymbol{s} = (s_0, s_1, \ldots, s_n)$, such that the concatenation of all the segments from $\boldsymbol{s}$ fully reconstructs the word w. Note that the length of a subword piece s_i can go from one character to all the characters of the word w. However, regardless of whether a subword piece represents a character in a word or not, all the pieces are treated as semantic units within a sentence.[3] This happens in current transformer-based models, where their strict tokenization into subword pieces has a significant impact on the semantic abstraction of upper layers of the model. A rule-based system generates the tokenization without taking any semantics into account [51]. This makes it hard to guarantee that the resulting tokenization is optimal for the model to

[3] [30] shows that BERT abstracts linguistic properties within its self-attention probabilities, evidencing that subwords need to preserve meaning when fed into such layers. This suggests that subwords broken down to individual characters can prevent the model from exploiting such linguistic properties at the sentence level.

interpret the underlying semantics of the sentence.

To mitigate such problems, I build word representations out of characters. The character-to-subword (char2subword) module allows flexible tokenization patterns, where the model can split by spaces, use the original tokenization method, or employ a different tokenization process as defined by the user. There are two main phases in the proposed method: approximating subword embeddings with the char2subword module (i.e., ideally replicating the embedding space $\boldsymbol{E}$) (Section 8.1.1), and contextually integrating the char2subword module into the pre-trained model (Section 8.1.2).

8.1.1 Approximating the Subword Embedding Table

Consider a subword s_i from the vocabulary $\mathcal{V}$ and a subword embedding matrix $\boldsymbol{E} \in \mathbb{R}^{|\mathcal{V}| \times d}$.[4] I learn a parameterized function $f_\theta : \mathbb{R}^{|c| \times 1} \rightarrow \mathbb{R}^d$ that maps the sequence of characters $\boldsymbol{c}_i = (c_{i1}, c_{i2}, \ldots, c_{i|s_i|})$ from the subword s_i to its corresponding embedding vector $\boldsymbol{e}_i \in \boldsymbol{E}$:

$$\hat{\boldsymbol{e}}_i = f_\theta(\boldsymbol{c}_i) \quad s.t. \;\; \hat{\boldsymbol{e}}_i \approx \boldsymbol{e}_i$$

To accomplish that, I optimize f_θ by minimizing the overall objective function $\mathcal{L}(\cdot)$ using the entries from the BERT's vocabulary $\mathcal{V}$:

$$\mathcal{L}(\boldsymbol{c}_i, s_i, \boldsymbol{e}_i, f_\theta) = \; \mathcal{L}_{cos}(\boldsymbol{e}_i, f_\theta(c_i)) + \mathcal{L}_{ce}(s_i, f_\theta(c_i)) + L^2(\boldsymbol{e}_i, f_\theta(c_i)) + \mathcal{L}_{nbr}(\boldsymbol{e}_i, f_\theta(c_i))$$

The first objective, $\mathcal{L}_{cos}(\cdot)$, is the cosine distance between the target and the predicted embedding vectors $\boldsymbol{e}_i$ and $\hat{\boldsymbol{e}}_i$. By using an angular distance function, I force the model to

[4]The subword embedding matrix $\boldsymbol{E}$ is part of the BERT model; it contains one-to-one mappings between the model's subword vocabulary $\mathcal{V}$ and the input subword embedding space.

replicate the semantic relationships and the vector arrangement in the embedding space $\boldsymbol{E}$:

$$\mathcal{L}_{cos}(\boldsymbol{e}_i, \hat{\boldsymbol{e}}_i) = \ 1 - cos(\boldsymbol{e}_i, \hat{\boldsymbol{e}}_i)$$

The second objective, $\mathcal{L}_{ce}(\cdot)$, is the cross-entropy loss function. In this case, I use $\boldsymbol{E}$ as fixed parameters (i.e., not trainable) to project linearly from the embedding to the vocabulary space. This loss term forces the model to learn accurate embedding representations such that they map to the original subwords from the vocabulary $\mathcal{V}$:

$$\mathcal{L}_{ce}(s_i, \hat{\boldsymbol{e}}_i) = -s_i \log p(\hat{\boldsymbol{e}}_i \cdot \boldsymbol{E}^\top)$$

The third objective is the L^2 norm or euclidean distance between the vectors $\boldsymbol{e}_i$ and $\hat{\boldsymbol{e}}_i$. The previous objectives do not regulate the magnitude of the predicted vector $\hat{\boldsymbol{e}}_i$, allowing that to be a degree of freedom for f_θ. By using the L^2 norm, I penalize the model for generating a vector $\hat{\boldsymbol{e}}_i$ with a different magnitude than $\boldsymbol{e}_i$. Note that preserving the magnitude is as important as preserving the vector arrangements in the embedding space. Intuitively, slightly different properties than the embedding space $\boldsymbol{E}$ can magnify differences at the upper layers of mBERT.

The last objective, $\mathcal{L}_{nbr}(\cdot)$, is the mean squared error (MSE) of cosine distances[5] generated between the k-th closest neighbors to $\boldsymbol{e}_i$ vs. the distances of the same neighbors for $\hat{\boldsymbol{e}}_i$:

$$(\boldsymbol{n}_1, \dots \boldsymbol{n}_k) = topk(\boldsymbol{e}_i, \boldsymbol{E})$$
$$\mathcal{L}(\boldsymbol{e}_i, \hat{\boldsymbol{e}}_i) = \frac{1}{k} \sum_{j=1}^{k} (dis(\boldsymbol{e}_i, \boldsymbol{n}_j) - dis(\hat{\boldsymbol{e}}_i, \boldsymbol{n}_j))^2$$

where $topk(\cdot)$ retrieves the k-th closest neighbors according to the cosine distances among all

[5]The cosine distance is defined as $dis(\boldsymbol{a}, \boldsymbol{b}) = 1 - cos(\boldsymbol{a}, \boldsymbol{b})$ where $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^d$.

the subword vectors in $\boldsymbol{E}$. The main idea of this objective is to force distances between $\hat{\boldsymbol{e}}_i$ and the neighbors $\boldsymbol{n}_*$ to be as similar as possible to the distances between the same neighbors and $\boldsymbol{e}_i$.

Char2subword module I model the char2subword module using the Transformer [116]. The module processes a sample as a sequence of characters $\boldsymbol{c}_i = (c_{i1}, c_{i2}, \ldots, c_{iM})$ of a subword s_i of length M.[6] I represent the sequence $\boldsymbol{c}_i$ as the sum between the character embeddings and sinusoidal positional encodings. I pass the resulting sequence of character vectors $\boldsymbol{X}_0$ to a stack of l attention layers, each with k attention heads. The j-th attention layer receives the input $\boldsymbol{X}_j$, and it outputs $\boldsymbol{X}_{j+1}$ by applying two subsequent components: multi-head attention and feed-forward layers. The multi-head attention is defined as follows:

$$\begin{aligned}
\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) &= \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d'}})\boldsymbol{V} \\
\text{MultiHead}(\boldsymbol{X}) &= [\text{head}_1; \ldots; \text{head}_k]\boldsymbol{W}^O \\
\text{where head}_i &= \text{Attention}(\boldsymbol{X}\boldsymbol{W}_j^{Q_i}, \boldsymbol{X}\boldsymbol{W}_j^{K_i}, \boldsymbol{X}\boldsymbol{W}_j^{V_i}) \\
\boldsymbol{X}'_j &= \text{MultiHead}(\boldsymbol{X}_j)
\end{aligned}$$

The feed-forward component linearly projects $\boldsymbol{X}'_j$ using $\boldsymbol{W}_{j1} \in \mathbb{R}^{d' \times 4d'}$ followed by a GELU activation function [52]. The projection is passed to another linear transformation such that the result $\boldsymbol{X}'_j$ is mapped back to $\mathbb{R}^{d'}$:

$$\text{FFN}(\boldsymbol{X}'_j) = \text{GELU}(\boldsymbol{X}'_j\boldsymbol{W}_{j1} + \boldsymbol{b}_{j1})\boldsymbol{W}_{j2} + \boldsymbol{b}_{j2}$$

Each component normalizes its input $\bar{\boldsymbol{X}}_j = \text{LayerNorm}(\boldsymbol{X}_j)$ using layer normalization [15].

[6] To distinguish between words and subwords, I prepend '##' to the sequence $\boldsymbol{c}_i$ in the case of full words.

I add the normalized input to the output of the component as in a residual connection [50]:

$$\boldsymbol{X}'_j = \text{MultiHead}(\bar{\boldsymbol{X}}_j) + \bar{\boldsymbol{X}}_j$$
$$\boldsymbol{X}_{j+1} = \text{FFN}(\bar{\boldsymbol{X}'}_j) + \bar{\boldsymbol{X}'}_j$$

Following [116], I preserve the dimension d' of the character embedding throughout the attention layers. On top of the l attention layers, I add a linear layer $\boldsymbol{W}_e \in \mathbb{R}^{d' \times d}$ followed by max-pooling and a layer normalization for the final output $\hat{\boldsymbol{e}}_i$:

$$\hat{\boldsymbol{e}}_i = \text{LayerNorm}(\text{maxpool}(\boldsymbol{X}_l \boldsymbol{W}_e + \boldsymbol{b}_e))$$

Character-level robustness The flexibility of the char2subword module makes it easier to teach the model text invariance because the inputs are now processed at the character-level. I augment the subword vocabulary $\mathcal{V}$ by introducing natural single-character misspellings during training. I apply one operation at a time and only to subwords that exceed four characters to reduce the chance of ambiguity between valid subwords. The operations are described in Table 8.1 and the high-level view of the approximation appears in Figure 8.3.[7]

8.1.2 Pre-training with the Char2subword Module

The previous techniques leverage the pre-trained knowledge in the embedding matrix $\boldsymbol{E}$. However, the char2subword module may not be appropriately integrated with the pre-trained mBERT's upper layers since it has only seen individual subwords without context. To alleviate that, I resume pre-training using the char2subword module along with the original parameters of mBERT [48]. Importantly, I do not update parameters in upper layers of

[7]For the `mistype` operation, I use over 100 keyboard layouts to cope with the languages in mBERT.

Table 8.1: Single-character operations to incorporate noise in the approximation stage. The operations are applied to every word in the vocabulary that exceeds the four characters and that it is not a special token.

Operation	Description
`mistype`	replace a random character of a given subword by randomly choosing from its nearby keys according to a keyboard layout
`repeat`	repeat a random character of a given subword
`swap`	randomly choose a character and swap it with the next character in the subword
`drop`	randomly drop one character of a given subword
`toggle`	toggle the case of a randomly chosen character from a given subword
`punctuation`	randomly insert a punctuation mark commonly used within text (e.g., parenthesis, dashes, periods, etc.)

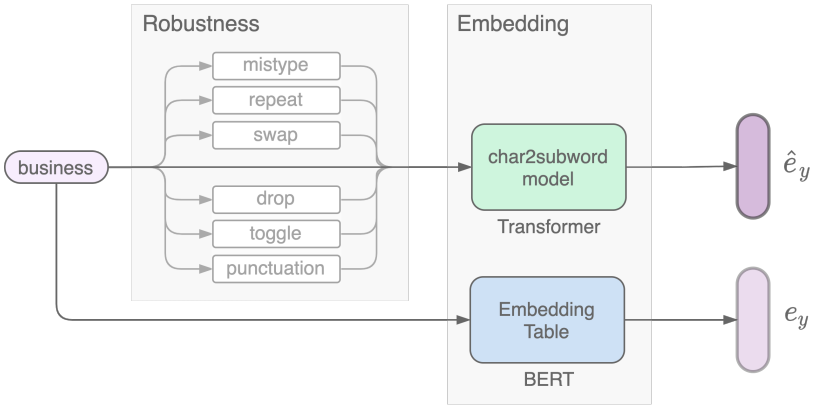


Figure 8.3: The char2subword module is trained using the subword embedding lookup table from a pre-trained language model (e.g., mBERT). I incorporate noise in every word at the character level with single-character operations.

mBERT since the goal is to provide the char2subword module as a drop-in alternative for $\boldsymbol{E}$ on the publicly available pre-trained checkpoint.[8]

Following practices from [67], I use a dynamic masked language modeling (MLM) objective and do not include the next sentence prediction objective (see Figure 8.4). I randomly choose 15% of the subword tokens and mask them at the character level. The masking

[8]While the study focuses on mBERT, this method can be applied to other multilingual (e.g., XLM-RoBERTa) or monolingual (e.g., CTRL [59]) models.

process replaces 80% of the characters with `[MASK]`, 10% with randomly chosen characters and the remaining 10% is left unchanged. I feed characters to the char2subword module and make predictions from the subword vocabulary $\mathcal{V}$.[9]

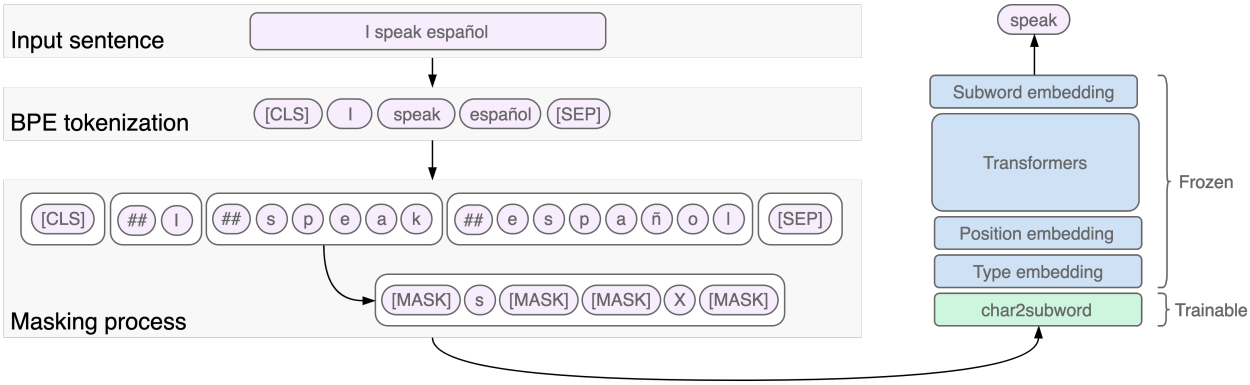


Figure 8.4: An example of an input and output of the pre-training setting with a masked language modeling (MLM) objective at the character level.

I pre-train the char2subword model with 1M sequences of 512 subword tokens from Wikipedia (200K sequences for each English, Spanish, Hindi, Nepali, and Arabic text). Using gradient accumulation, I update parameters with an effective batch size of 2,000 samples. Note that the model does not require extensive pre-training since 1) the upper-layer parameters are initialized from the pre-trained mBERT checkpoint and kept fixed during training, and 2) the char2subword module is initialized from the embedding approximation phase. Thus, pre-training the model for a few epochs is sufficient.

8.1.3 Fine-tuning

Once the char2subword module has been optimized, I evaluate the pre-trained model with the char2subword module on downstream NLP tasks. Specifically, I experiment with two scenarios: the full and the hybrid modes.

[9]Using the pre-trained embedding table $\boldsymbol{E}$, I project the internal representations per word onto the vocabulary space without updating the parameters of $\boldsymbol{E}$.

Full mode This mode completely replaces the subword embedding table in mBERT (i.e., the set of parameters and vectors) with the char2subword module (see Figure 8.5 (center)). This setting's idea is to evaluate how well approximated was the embedding space originally in $\boldsymbol{E}$. Intuitively, if the char2subword replicates the embedding space in $\boldsymbol{E}$ perfectly, then the overall model should behave about the same as the original mBERT model. Nevertheless, this setting does not tokenize a word further; hence, the input sequence tends to be shorter and more meaning-preserving (i.e., too many subword pieces for a single word can degrade its meaning). This setting should overcome misspellings while also preserving the original knowledge in the embedding table without splitting subwords dramatically into multiple pieces.

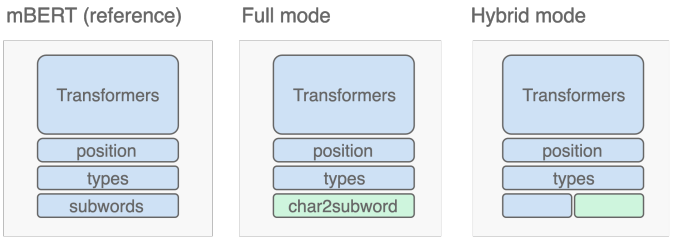


Figure 8.5: The fine-tuning scenarios with the char2subword module. The mBERT model structure (left) is added for reference. The full mode (center) shows the char2subword module instead of the subword embedding table. The hybrid mode (right) represents both the subword embedding table and the char2subword module. The char2subword module is only used when the input word does not appear in the vocabulary as a whole.

Hybrid mode Unlike the full mode, this mode does not replace the subword embedding table (see Figure 8.5 (right)). Instead, it uses the subword embedding vectors by default for full words (i.e., not subword pieces). The model backs off to character-based embeddings from the char2subword module when a word as a whole does not appear in the vocabulary. This method prevents words from being broken down into pieces by building word representations out of its characters.

8.2 Experiments

8.2.1 Embedding Approximation

The approximation experiments' goal is to replicate the original subword embedding table while ensuring robustness at the character-level modifications. I experiment with the objective functions described in Section 8.1.1. I use the average precision to determine the best method. Nevertheless, I also provide the accuracy for reference.[10]

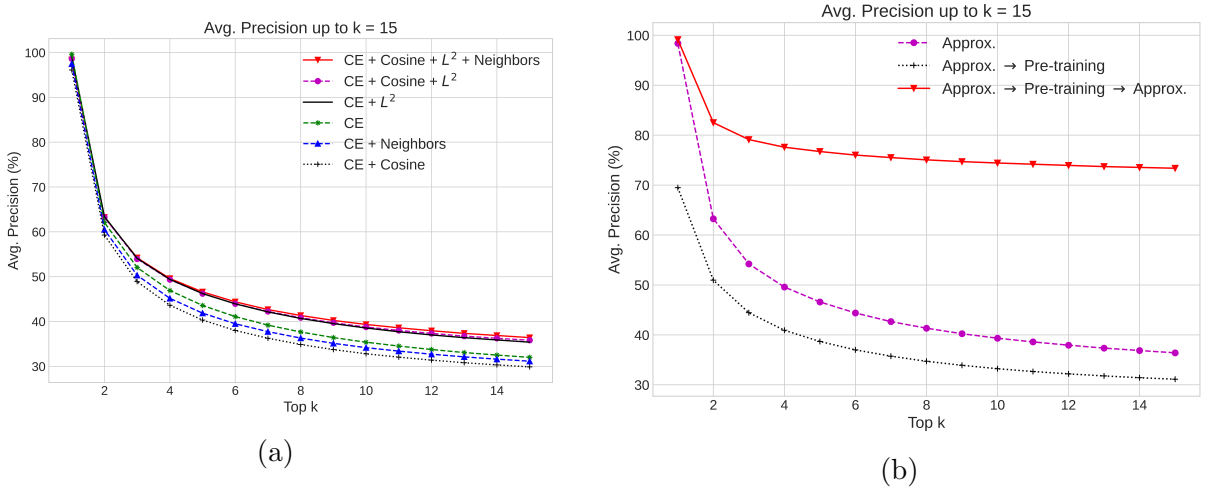
The experiments 1.1 to 1.4 show the results of each objective function individually (see Table 8.2). Notably, the cross-entropy objective is the most relevant to ensure high precision (58.1% vs. 28.5% of the cosine objective from experiments 1.1 and 1.2, respectively). Combining all the objective functions as in experiment 1.9 gives an average precision of 60.0%. Although experiment 1.6 performs very close to experiment 1.9 (59.9% vs. 60.0%), experiment 1.9 still preserves more neighbors along the top k expected neighbors (see Figure 8.6a).

After optimizing a char2subword module in experiment 1.9, I contextualize it according to the pre-training phase (Section 8.1.2). The results show that the precision at k drops substantially (see "Approx. $\rightarrow$ Pre-training" in Figure 8.6b). However, when restoring approximation from the pre-training phase, the model performs far better than the approximated version reaching an average precision of 82.4% (see "Approx. $\rightarrow$ Pre-training $\rightarrow$ Approx." in Figure 8.6b). This bounce in performance shows the need for contextualization for the original char2subword module. The contextualization by itself does not guarantee that the module will resemble the same embedding space as in $\boldsymbol{E}$ (i.e., there is nothing that forces the module to optimize for that). However, it provides a better arrangement of the

[10]Using accuracy to determine the best method can mislead the interpretation of the model's capabilities. Accuracy is not ideal in this scenario since the goal is to approximate an embedding space rather than merely predicting vocabulary subwords given their characters. To better assess the embedding space approximation, I use average precision up to the top k neighbors of $\boldsymbol{e}_i$.

Table 8.2: The results of approximating the subword embedding table from mBERT using different combinations of objective functions. Experiments 1.1 to 1.4 denote the performance of the char2subword module using individual objective functions (e.g., experiment 1.2 only uses $\mathcal{L}_{cos}$). Experiments 1.5 to 1.8 use the cross-entropy objective $\mathcal{L}_{ce}$ by default and combine it with other objectives (e.g., experiment 1.8 uses $\mathcal{L}_{ce}(\cdot)+\mathcal{L}_{neigh}(\cdot)$). Experiment 1.9 combines all the objectives at the same time. The accuracy denotes the capability of the model to predict a subword out of its characters. Precision @ k measures the overlap between the k ground-truth neighbors for a vector $\boldsymbol{e}_i$ (that represents subword s_i) and the k neighbors of the predicted vector $\hat{\boldsymbol{e}}_i$.

Exp. ID	$\mathcal{L}_{ce}$	$\mathcal{L}_{cos}$	L^2	$\mathcal{L}_{nbr}$	Accuracy	Prec@1	Prec@15	Avg Prec
1.1	✓				99	99.6	43.9	58.1
1.2		✓			62	41.8	24.2	28.5
1.3			✓		45	18.2	12.2	13.5
1.4				✓	43	25.5	17.1	19.6
1.5	✓	✓			96	96.1	41.2	55.1
1.6	✓		✓		95	99.1	46.6	59.9
1.7	✓	✓	✓		95	98.6	46.7	59.8
1.8	✓			✓	98	97.4	42.6	56.5
1.9	✓	✓	✓	✓	95	98.3	47.1	**60.0**



(a) (b)

Figure 8.6: Precision up to the 15-th top elements. (a) A comparison of the most important objective functions for the approximation phase. (b) A comparison of the best approximation model before and after the pre-training stage.

embedding space, which results in better average precision in the approximation.

Character-level robustness is another essential aspect when optimizing the char2subword module. I add single-character perturbations to the training phase as mentioned in Section

8.1.1. Table 8.3 shows the neighbors of the word *business* and its variations. Note that when the input is *business* without perturbations (first column), the char2subword modules (with and without noise) retrieve semantically-related neighbors. However, when the word is capitalized, the neighbors are not related to the word *business* for the char2subword without noise. Also, the subword tokenization for *BUSINESS* becomes *B-US-INE-SS*, which undermines the meaning of the original word. Regardless of that, the char2subword noise is resilient to the capitalization pattern and capable of maintaining the meaning.[11]

Table 8.3: The results of approximating the subword embedding table from BERT. The first row describes the neighbors of the query vectors coming from the same embedding table $\boldsymbol{E}$. Note that for words that do not exist in the table, I only show how the word would be tokenized. The other rows contain neighbors in every cell since they always have representations for the sequence of characters.

Model	**Input Tokens**			
	business	**BUSINESS**	**busINess**	**bsusinessses**
mBERT Subword Table	business (1.00) Business (0.61) businesses (0.47) businesses (0.47) bisnis (0.46)	*B-US-INE-SS*	*bus-IN-ess*	*b-sus-iness-ses*
Char2subword	business (0.82) Business (0.50) businesses (0.43) bisnis (0.38)	ASEAN (0.2565) RSS (0.2456) FCC (0.2416) WEB (0.2403) Austrália (0.2360)	pisos (0.22) islas (0.21) tienda (0.20) Conservatory (0.20) Filipinas (0.20)	businesses (0.42) companies (0.33) opportunities (0.32) industries (0.31)
Char2subword + Noise	business (0.80) Business (0.61) businesses (0.53) negocios (0.39)	Business (0.53) business (0.32) Marketing (0.31) Corporate (0.31) Communications (0.30)	business (0.79) Business (0.61) businesses (0.53) companies (0.38)	businesses (0.79) companies (0.53) shops (0.52) corporations (0.50) employees (0.49)

[11] Interestingly, the char2subword module never sees a subword from the vocabulary with more than a single character edit (i.e., I defined the robustness procedure this way). That means that the word *BUSINESS* never appeared in training for the model.

8.2.2 Fine-tuning Experiments

Once the char2subword module is adapted to mBERT, I benchmark the model in the full and hybrid modes (see Section 8.1.3) using the linguistic code-switching evaluation (LinCE) benchmark [2]. In particular, I focus on the sequence labeling tasks from LinCE: language identification (LID), part-of-speech (POS) tagging, and named entity recognition (NER).

Table 8.4 shows the results of the experiments using the full and hybrid modes. For each model, I use the approximated and pre-trained (i.e., "Approx. $\rightarrow$ Pre-training $\rightarrow$ Approx.") versions of the char2subword module. The language identification results are not a strong indicator of improvement since the scores are all very close.[12] Nevertheless, it is important to note that the model can perform on par with the mBERT baseline regardless of the version. This suggests that the char2subword module is learning good representations compatible with the rest of the mBERT model (i.e., mBERT transformer layers).

For the POS and NER tasks, the tendency is different. The hybrid pre-trained experiment for Hindi-English is significantly better than the baseline for both POS (89.64% vs. 87.86%) and NER (74.91% vs. 72.94%). One of the reasons for this performance boost is due to the noise that splitting transliterated Hindi (i.e., Hindi wrote with the Roman script) generates for the baseline. On the contrary, the char2subword compresses the transliterated words into a single vector, reducing the model's noise.

The NER results for Spanish-English (es-en) and Modern Standard Arabic-Egyptian Arabic (msa-arz) also exceed the baseline (64.26% vs. 62.66%). Although there is no transliteration in these language pairs, there is still much noise coming from social media user-generated language. Also, pre-training the char2subword on Spanish and Arabic data improves the model's representations and robustness for such languages.

Table 8.5 shows consistent results with the development set. The LID results are not

[12]The average score for LID across language pairs is 95.71% for mBERT (baseline) and 95.80% for char2subword module (hybrid, pre-trained).

Table 8.4: Results on the development set of the LinCE benchmark (average over three runs with different seeds). Full refers to the full mode where the model only uses the char2subword to embed the input. Hybrid means that the model uses the subword embedding table by default and backs off to the char2subword module for unseen words (i.e., out-of-vocabulary words) instead of splitting it. For this table, pre-trained means that the model was approximated after the pre-training phase (i.e., "Approx. → Pre-training → Approx."), while approx. means that the model is only trained in the approximation phase. The languages involved are English (en), Spanish (es), Hindi (hi), Nepali (ne), Modern Standard Arabic (msa), and Egyptian Arabic (arz). The scores for LID, NER, and POS are weighted F_1, micro F_1 with spans, and accuracy, respectively. The best results on each language pair are in bold.

			LID (W. F_1)				**POS (Acc.)**		**NER (F_1)**		
Method	**Adaptation**	**Avg**	es-en	hi-en	ne-en	msa-arz	es-en	hi-en	es-en	hi-en	msa-arz
mBERT	N/A	86.95	98.23	96.37	**96.67**	91.55	**97.29**	87.86	62.66	72.94	78.93
Full	Approx.	86.66	98.16	95.79	96.45	91.63	96.93	89.04	62.02	70.79	79.13
Full	Pre-trained	86.89	98.20	96.97	96.47	91.48	96.91	89.38	61.23	71.98	79.42
Hybrid	Approx.*	87.37	**98.24**	**96.98**	96.50	91.48	97.16	88.95	**64.26**	72.68	80.10
Hybrid	Pre-trained*	**87.59**	98.18	96.75	96.37	**91.64**	97.03	**89.64**	63.32	**74.91**	**80.45**

* Statistically significant with respect to the mBERT baseline, with p-value < 0.01 in student's t-test.

significantly different, but the results on the POS and NER tasks are.

Table 8.5: Results on the test set of the LinCE benchmark using the hybrid char2subword mBERT model (best proposed model). The languages involved are English (en), Spanish (es), Hindi (hi), Nepali (ne), Modern Standard Arabic (msa), and Egyptian Arabic (arz). The scores for LID, NER, and POS are weighted F_1, micro F_1 with spans, and accuracy, respectively. State-of-the-art performance reached as of October 19th, 2020: `https://ritual.uh.edu/lince/leaderboard`.

		LID				**POS**		**NER**		
Method	**Avg**	es-en	hi-en	ne-en	msa-ea	es-en	hi-en	es-en	hi-en	msa-ea
mBERT	85.09	98.36	94.24	96.32	91.55	97.07	86.30	64.05	72.57	65.39
Best proposed model	**85.69**	98.33	96.23	96.19	91.19	96.88	88.23	64.65	73.38	66.13

8.3 Analysis

Attention for LID Figure 8.7 shows the visualization for the Spanish-English LID task with an intra-sentential code-switching example (i.e., code-switching at the clause level of

a sentence utterance). The example shows that the strongest connections at the word level (Figure 8.7 (left)) happen for words in the same language. Particularly, the word *consequencias* is slightly ambiguous since its morphology overlaps substantially with both the English and Spanish versions. With the context from the surrounding Spanish words, the model can determine that the word is Spanish. Although there are more patterns captured among all the heads in mBERT, this pattern suggests that words of the same language can provide contextual support along with the sentence.

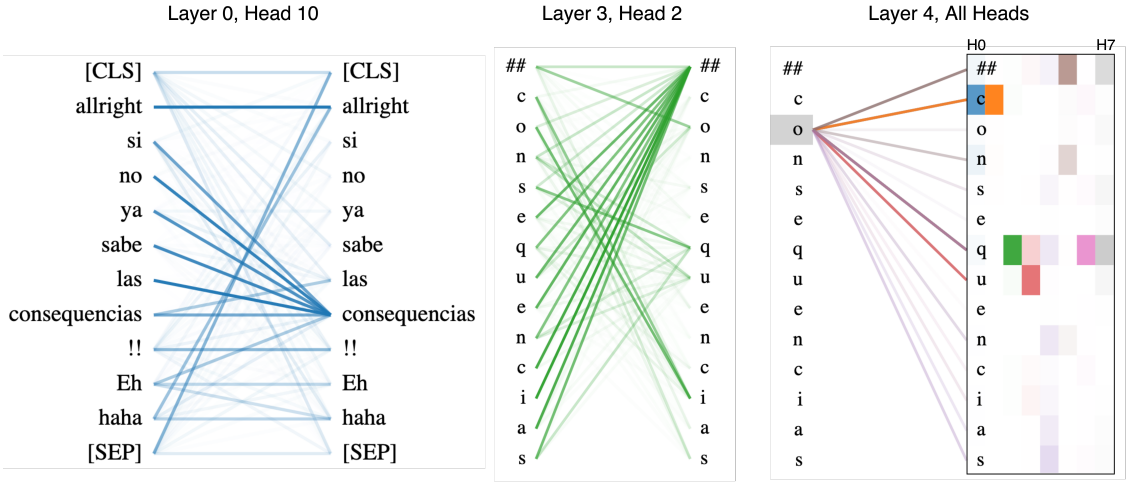


Figure 8.7: Character-level attention for a Spanish-English tweet. The connections between words read from left to right, and they represent the probability mass distributed across the sentence for one word (i.e., a self-attention head). The right figure shows the attention distribution across the eight heads (i.e., from H0 to H7). Translation: *"Alright, otherwise you know the consequences!! Eh, haha."*

In addition to the contextual support, character-level attention (i.e., char2subword transformer layers) also plays an important role when building the word representation. Particularly for this word, the ambiguity is introduced due to the letter *q*. Note that the char2subword module creates strong connections with this letter and parts where more ambiguity could happen. For example, the letter *i* in the word *consequencias* happens where the suffixes *-cias* (Spanish) and *-ces* English could complete the word).

Error analysis Table 8.6 captures the mistakes of the model in a confusion matrix for the Spanish-English LID development set. Note that there are many mistakes between the English and Spanish words (112 English words predicted as Spanish, and 101 Spanish words predicted as English). Out of the 101 English words, 63 were processed by the char2subword module (i.e., via back-off). Most of these errors come from words that heavily overlap in morphology between the two languages. For example, the words *imagine, rodeos, superego, tacos* are exact spellings between the languages, while the words *apetite* and *pajamas* change one letter between the languages (e.g., *apetito, pijamas* in Spanish). These errors suggest that the robustness may create some ambiguity when detecting the text's language. That is, single-character differences can denote one or another language, but the robustness operations (Table 8.1) can blur such distinction during the approximation phase. Other words are interjections that are spelled the same way (e.g., *oh, eh, Muahahahahaha*). Also, there are cases where the ground-truth labels are wrong. For example, the word *larges* in the the sentence "*La puerta está abierta para que te larges, ¿por qué no te has ido?*"[13] was correctly predicted as Spanish based on the context (i.e., the correct spelling is *largues*, which translates to *get out*).

Lastly, the mixed LID class shows that words code-switched at the surface level are difficult to detect (the model obtains zero predictions out of six tokens in the validation set). Although this is a deficient number of instances, the model shows a particular aspect across classes. The model tends to rely more on the semantic meaning of the word rather than on its spelling. For example, the word "taggea" is an English word conjugated with the Spanish grammatical rules, and the model predicts it as a Spanish word. The word "estop" can be *stop* or *estop* but either way, they have the same meaning, and the model relies more on the context than on the ambiguous spelling, predicting it as an English word. This could

[13]Translation: "The door is open for you to get out, why haven't you left?"

be explained by the robustness emphasis (e.g., character edits from the multilingual space) while approximating the embedding space.

Table 8.6: The confusion matrix on the development set of the LID task for Spanish-English. The labels are lang1 (English), lang2 (Spanish), mixed (partially in both languages), ambiguous (either one or the other language), fw (a language different than lang1 and lang2), ne (named entities), other, and unk (unrecognizable words).

	Ground-truth							
Predictions	**ambiguous**	**fw**	**lang1**	**lang2**	**mixed**	**ne**	**other**	**unk**
ambiguous	0	0	21	16	0	0	1	1
fw	0	1	0	1	0	0	0	0
lang1	14	0	16,492	101	0	74	14	17
lang2	13	0	112	14,771	0	51	5	3
mixed	0	0	1	4	0	1	0	0
ne	3	0	110	96	1	597	7	1
other	1	0	13	6	1	3	7,802	4
unk	0	0	8	10	0	3	3	8

The errors in transliterated language have different patterns. The char2subword module attempts to generate embeddings that resemble the morphological form of the input words. Hence, the module works as an extension of the morphological patterns in the original vocabulary (in addition to variations with noise described in Section 8.1.1). In Hindi-English language identification, the char2subword module has a more substantial bias towards English (i.e., lang1) than transliterated Hindi (i.e., lang2), particularly for words whose spellings overlap between the two languages—the module is not pre-trained on transliterated Hindi. For example, the tweet *"I do not see the left and right. Dazzled my eyes becch mein baithe,"*[14] the word *becch* is misspelled and could be either *beach* or *beech* (*beech* translates to *middle*). Since the char2subword module has no access to the context at the point of building the initial word embedding vector, it cannot disambiguate the spelling and retrieve a more informed representation. This causes the word to be wrongly labeled as English instead of

[14]Translation: *"I do not see the left and right. Dazzled my eyes sitting in the middle."*

Hindi. A similar scenario happens with Hindi words like *yug* (meaning *era*) and *par* (meaning *on*), which show neighbor embeddings like *young* and *pair*, respectively. Unlike many cases in Spanish-English, similar spellings between Hindi and English do not usually have similar meanings.[15] Therefore, spelling ambiguity is more harmful in such scenarios because the meaning can entirely change.

Another interesting behavior in the Hindi-English language identification predictions is the tendency of detecting single-letter words as Hindi. Since there is no official agreement for Hindi transliteration with the Roman script, code-switchers employ the Roman script relying on the sounds as per its English pronunciation. For convenience, they also use the shortest possible ways to write such sounds on social media. For example, frequent single-letter words like *g*, *m*, *h*, and *k* can be spelled more adequately as *ji*, *hum*, *he*, and *ke* to align better with their pronunciation. Nevertheless, these aspects make the model pick up single-letter words more often as Hindi than English without the model knowing the underlying reason (i.e., pronunciation), often resulting in wrong predictions where there is a lack of English context.

Subword sequence lengths Naturally, sequences coming from the subword tokenization are at least the same length or longer than the original sequence of tokens. Quantifying that aspect shows the opportunity that the char2subword mBERT model has in practice. Table 8.7 shows the statistics of the original sequence lengths (Tokens) and the sequence lengths after the subword tokenization (Subword). Note that the average sequence lengths tend to duplicate across datasets. This can potentially explain a larger gap in performance for NER and POS tagging tasks than in LID. The former tasks require more semantics, which aligns with the fact that subwords degrade meaning by splitting into many pieces.

[15] Unless words are adopted from one language to the other, like *rupees* or *lakhs* (i.e., hundred).

Table 8.7: Statistics across the development sets that compare sequence lengths before and after subword tokenization. **Tokens** refers to the original length of the sequences as provided in the benchmark, while **Subwords** means the same sequence further tokenized with the BPE algorithm employed in multilingual BERT.

			Tokens			**Subwords**		
Task	**Lang.**	**Samples**	**Mean ± Std**	**Min**	**Max**	**Mean ± Std**	**Min**	**Max**
LID	es-en	3,332	12.1 ± 7.7	1	39	21.1 ± 12.0	1	69
	hi-en	744	20.8 ± 24.1	1	225	31.4 ± 32.9	4	278
	ne-en	1,332	14.5 ± 6.3	3	34	28.5 ± 10.8	3	63
	msa-arz	1,116	19.7 ± 6.5	2	36	43.5 ± 14.4	2	93
NER	es-en	10,085	12.1 ± 7.6	1	45	25.7 ± 14.2	1	120
	hi-en	314	17.0 ± 6.3	4	34	40.5 ± 13.6	7	74
	msa-arz	1,122	20.2 ± 6.7	2	38	44.5 ± 14.8	3	112
POS	es-en	4,298	7.7 ± 6.0	2	90	9.9 ± 7.8	2	127
	hi-en	160	21.7 ± 5.2	5	37	41.3 ± 12.2	7	93

Parameters vs. efficiency The subword lookup table in mBERT provides immediate access for the tokenized text to the embedding space, making such a table very convenient. However, this access is highly restricted to a predefined vocabulary, and, in the case of multilingual models, such vocabulary has to have adequate coverage for all the languages involved. Models like mBERT or XLM-RoBERTa use more than 100 languages, which translates into a large number of parameters just to enable the text to be vectorized. More specifically, mBERT has 177M parameters in total while only its subword embedding table ($|\mathcal{V}| = 119K$) occupies 91M parameters—more than 50% of all the parameters of the model.[16] The char2subword module, on the other hand, reduces the number of parameters to 50M, about 45% less than the subword embedding table, while also capable of handling misspellings and inflections robustly. Nevertheless, this module requires more computation time to come up with subword-level embedding representations.

[16]For XLM-RoBERTa base (278M) and large (559M), the percentages are 65% and 49%, respectively.

8.4 Limitations

The main limitations of this work are related to measuring the quality of the approximated embeddings. Extending the multilingual embedding space is a sensitive task, and producing those embeddings are crucial for the model to have good performance. However, I only measure the quality by checking neighbors in cosine-distance space. Although useful and reliable for the neighbors, this metric does not necessarily capture all the properties of the embedding space. The metric focuses on each embedding vector subspace's localities, but not necessarily on the global aspects.

Another limitation is the cost of the pre-training phase to contextualize the char2subword module. Even though this phase only updates parameters on the char2subword module (not the overall multilingual BERT model), it still requires extensive computational resources[17] The computing demand reduces the scope for exploring multiple hyper-parameters, limiting this part of the work to follow suggested practices.

The last major limitation is related to pre-training with the subword tokenization as part of the process. The overall model (e.g., mBERT) uses a masked language model objective over its original vocabulary. Since the char2subword module takes advantage of the pre-training, the same subword parameters are used to predict the output (e.g., the embedding matrix $\boldsymbol{E}$, meaning that a vocabulary does not have to be learned). That induces subword units into the char2subword module even though the model could entirely use a different tokenization process (e.g., as simple as splitting the text by spaces). This is not an ideal outcome because subword units degrade meaning, and using poor-meaning pieces as input can deteriorate the overall model performance. Ideally, to drop subword tokenization entirely, the model should be pre-trained from scratch using inputs as characters for every word (i.e.,

[17] I conducted the pre-training phase using 32 GPUs Tesla V100, with 1M training examples. I trained the model for four days on four epochs.

character-level feature extraction), reducing them into a single vector per word (e.g., through max-pooling operations). Then, the masked language modeling objective could be applied at the character level in a one-to-many mapping (i.e., from a word to its characters). This would be ideal because the loss would penalize the model by both word meaning and morphological inflection (e.g., the model does not have to learn a plural word different independently from its singular version).

8.5 Conclusion

This chapter provides a novel and flexible method to expand the subword embedding table from a model (e.g., mBERT). I conducted the study in this chapter with the case of linguistic code-switching and sequence labeling tasks. However, this method is not limited to code-switching or sequence labeling. The char2subword module provides more control at the tokenization level, and it can generate word embeddings without being restricted by a fixed vocabulary. Additionally, the module opens up the possibility to refine a language of interest, as shown in the approximation phase (e.g., by pre-training just the char2subword module). Finally, I showed the effectiveness of this method by reaching a new state of the art performance on the LinCE benchmark.

Chapter 9

Conclusions

This dissertation focused on the challenges that social media (SM) text poses to natural language processing (NLP) systems. In general, I proposed approaches that deviate from traditional text normalization methods that mitigate noise. I argued that the text's noise is part of what SM users want to convey through non-standardized language, and text normalization can inadvertently remove language-evolving properties. Besides, since SM platforms are intrinsically multilingual, noise can also be present in multilingual forms, such as linguistic code-switching, where text normalization techniques do not scale. Therefore, it is crucial to embrace user-generated noise as an inherent characteristic of SM language.

This dissertation covered sound-driven writing, flexible grammar, and arbitrary spellings for English and code-switched text on sequence labeling tasks. I describe the English and code-switching methods with their contributions in separate sections and conclude with notes on future work.

9.1 English-oriented Methods

In Chapters 3, and 4, I proposed methods motivated by phonological writing and dependency parsing for named entity recognition (NER). In the case of phonological writing, I showed that a model relying on sound-driven properties of SM text could more concisely capture the underneath semantics. Notably, the proposed system builds word representations out of the international phonetic alphabet (IPA). Based on the word's sound, a word is fed into the model as a sequence of IPA characters instead of using the English alphabet. This model outperformed previous systems that rely on gazetteers and normalization or use the English alphabet instead of IPA, proving the effectiveness of this model.

For dependency parsing, I used dependency trees to capture the semantic relationships among words. The motivation was that word relationships can disclose information about entities (e.g., the person class highly relates to verbs like speak, run, live, etc., while the entity class location relates less to these verbs). In addition to recurrently processing language from a straight sequence of words (i.e., LSTM), I proposed to model text in a hierarchical structure, guiding the processing by dependency trees (i.e., TreeLSTM). I experimented with these methods on newswire text (i.e., SemEval-2010 and CoNLL-2003) and social media text (i.e., WNUT-2016 and WNUT-2017). Importantly, the TreeLSTM model outperforms the LSTM counterpart when the trees' quality is guaranteed (i.e., no mistakenly connected nodes or an incorrectly segmented tree into many isolated trees). While this is the case for manually-annotated trees as in SemEval-2010, the TreeLSTM is not more effective than the LSTM in SM text. SM dependency parses are automatically generated, making them very error-prone due to the seemingly disconnected utterances. The resulting multi-rooted trees reduce the context to each subtree's words, isolating different parts of the text.

In Chapter 5, I investigated the effect of highly-frequent entities on English text. I experimented with state-of-the-art models like BERT and showed that the model could memorize

entities that appear during fine-tuning or leverage pre-training knowledge to determine an entity. This behavior was more prominent on newswire data because it highly resembled the pre-training data domain, considerably overlapping entity instances. Unlike newswire text, SM was less prone to leverage entities since they hardly match the spellings or context and are likely to be emerging and novel entities. I proposed methods that can alleviate the tendency to memorize frequent entities during fine-tuning as it was usually the case for the SM data. The procedures improve the model's generalization capabilities beating the vanilla version of BERT while keeping the techniques convenient and practical.

9.2 Code-switching-oriented Methods

In Chapter 6, I proposed the Linguistic Code-switching Evaluation (LinCE) benchmark. LinCE contains ten code-switching datasets, four tasks, and four language pairs. I examined each dataset and fixed significant issues on the partitions with a comprehensive stratification method. I also conducted the annotation of the Spanish-English text for named entity recognition (NER) and sentiment analysis. As of November 2nd, 2020, the LinCE platform has registered more than 50 users, 68 data downloads from personal accounts, 490 data downloads from the HuggingFace `datasets` library[1], 33 submissions to the leaderboard from which 9 are publicly ranked. LinCE aims at providing a centralized and unified benchmark for code-switching evaluation. This benchmark will serve as a transparent and efficient way of comparison on code-switching methods.

Besides, I proposed methods for linguistic code-switching applicable to text with the original scripts of the language pairs and transliterated text from one language script to the other. More specifically, I focused on character-level approaches that leverage pre-training

[1] `https://github.com/huggingface/datasets`

from large language models (e.g., ELMo and BERT). I experimented using pre-trained monolingual knowledge (e.g., ELMo) and, based on the effectiveness of this method, I extended the pre-trained multilingual knowledge of BERT to the code-switching scenario.

In Chapter 7, I introduced a code-switching-adapted ELMo (CS-ELMo). The motivation was that a monolingual model could tell apart its primary language (i.e., English) from an unknown language by exploiting morphology. Hence, I proposed a hierarchical attention component at the convolutional characters within ELMo. This method exploited the language pairs' morphological aspects fed to ELMo while adapting to the code-switching scenario. The model learned to detect morphological patterns that are uncommon in its primary language (e.g., Spanish infinite verb endings like *-ar* are not as common in English, while the gerund ending *-ing* hardly appears in Spanish). This adaptation step also shows its benefits on code-switching downstream tasks like NER and part-of-speech (POS) tagging for Spanish-English and Hindi-English data.

In Chapter 8, I introduced the character-to-subword (char2subword) module as an alternative for the subword embedding table in multilingual BERT (mBERT). While exploring mBERT in code-switching data, particularly transliterated text, the byte-pair encoding algorithm tended to break words into too many subword pieces. This behavior sensibly degraded the sentences' semantics because the input could come down to characters while mBERT expected word-level representations. Having seen the effectiveness of leveraging morphology in CS-ELMo with character-level components, the char2subword module aims at replicating the original subword embedding space while being more flexible to the spellings. The module showed that it was possible to extend the original vocabulary while generating high-quality word representations compatible with the subword embedding space. I showed the effectiveness of this module empirically using downstream tasks from the LinCE benchmark. As of November 2nd, 2020, the method is currently state of the art for sequence labeling tasks.

9.3 Detailed Contributions

Below, I consolidate the specific contributions mentioned across the chapters of this dissertation.

- **Chapter 3 - Sound-driven writing:**

    1. Removing dependencies on feature engineering that require constant updates for social media NER (e.g., gazetteers) by proposing a method that relies on automatic feature extraction from sound-driven writing. This method was the state-of-the-art by the time of publishing.

    2. Modeling sound-driven writing through articulatory (e.g., tongue postures, mouth movement, and face gestures) and phonetic (e.g., International Phonetic Alphabet) features that resemble social media patterns rather than task-specific aspects. The articulatory and phonetic features align with the flexible spelling that consistently preserves the sound-driven writing from social media users.

    3. Exploring a two-stage modeling method where the first model extracts features at the token level in a multi-task learning setting, while the second model optimizes the global sequence labels from the features of the first model. This method shows slight improvements over an end-to-end method, but more exploration is required.

- **Chapter 4 - Flexible syntax:**

    1. Showing empirically that recursively modeling language according to linguistic structures, such as dependency trees (i.e., Tree-LSTM), is more effective than sequential processing for NER. This finding is subject to good quality dependency trees (e.g., no multi-rooted trees or mistakenly connected nodes). The recursive

modeling results are better on long sequences since the hierarchical structure makes the model more resilient to long dependencies across the text.

2. Providing evidence that word connections yield substantial clues about the entity classes. Dependency trees connect words semantically, allowing the models to have immediate access to words that disclose explicit information about entities (e.g., the verbs *speak* or *think* are strongly associated to the entity class *person*).

3. Exploring different levels of attention mechanisms to leverage semantic connections across words. The proposed relative attention allows information to flow across the text while preserving the linguistic structure from the Tree-LSTM model. This attention mechanism captures more patterns from a Tree-LSTM than an LSTM, suggesting that hierarchical modeling projects more linguistic features than sequential processing.

- **Chapter 5 - Entity memorization vs. generalization:**

1. Providing insights from data analysis regarding entity frequency and its influence across partitions on the CoNLL-2003 and WNUT-2016 datasets. I showed that relying on the test set score to drive progress on NER is not ideal. The performance can be largely inflated by frequently seen entities in the training and development sets.

2. Inspecting a state-of-the-art model behavior on the CoNLL-2003 and WNUT-2016 NER datasets for unseen and seen entity instances, as well as the impact of entity frequency at different frequency thresholds. I showed that the models' performance tends to correspond directly to the number of seen entity instances. Nevertheless, the high frequency of entity instances has little impact on the performance as long as the entity class of such entity instances is diverse.

3. Proposing methods motivated from the insights and take-aways on the data analysis and model inspection. I proposed different methods to alleviate the bias towards memorization. The most effective method was a higher penalty loss on infrequent entities.

- **Chapter 6 - LinCE benchmark:**

1. Releasing a centralized and publicly available benchmark for the code-switching community to drive progress on ten datasets, four language pairs, and four tasks. The languages are Spanish-English, Hindi-English, Modern Standard Arabic-Egyptian Arabic, and Nepali-English. The tasks are language identification (LID), named entity recognition (NER), part-of-speech (POS) tagging, and sentiment analysis (SA). This benchmark alleviates the difficulty of comparing systems known effective for a particular language pair or task, but their effectiveness does not necessarily translate to other languages or tasks.

2. Conducting the annotation of the Spanish-English data for the named entity recognition and sentiment analysis (SA) tasks. These datasets are part of the LinCE benchmark.

3. Providing more reliable splits with a novel stratification technique. Even though providing new splits prevents comparison with work before LinCE, the original splits had acute problems (e.g., different label distributions and no label instances in a split). The proposed stratification considers sequence length, label distribution, and the language identification labels for tasks like NER, POS, or SA. The method provides consistent code-switching splits, reducing the KL-divergence among the training, validation, and test sets. This is important to allow researchers to project their performance from the validation set to the test set more

reliably due to the distribution consistency across partitions.

- **Chapter 7 - From English to code-switching:**

  1. Exploiting morphological patterns to detect between English and non-English languages by relying on pre-trained English knowledge. A pre-trained model like ELMo can detect English morphology due to its character convolutions. However, adapting itself to a code-switching setting is hard because the model has not seen other languages. I proposed CS-ELMo, which can detect what is and what is not English based on morphological abstraction.

  2. Introducing the enhanced character n-gram module within ELMo's character convolutions. This novel component allows ELMo to adapt to code-switching scenarios better than the vanilla ELMo architecture. That is due to the specific emphasis on morphology using hierarchical attention mechanisms. The attention is applied at every n-gram order separately and then across n-gram orders while considering the n-gram positions (e.g., the tri-gram *-ing* is a very common ending in English, but rare in Spanish).

  3. Proving downstream benefits on tasks like Spanish-English NER and Hindi-English POS tagging once the model has been adapted to the code-switching setting using language identification. The experiments showed that the adaptation of CS-ELMo to the code-switching setting improves performance off-the-shelf.

- **Chapter 8 - From multilingualism to code-switching:**

  1. Extending the multilingual subword embedding space using robust character compositionality. Based on insights from the previous chapter description, I designed the char2subword module to extend the initially restricted vocabulary space from

multilingual BERT using character-based representations. The module is compatible with the subword embedding space and allows out-of-vocabulary words to be handled without splitting words into pieces.

2. Improving multilingual sentence representations by reducing word splitting and meaning degradation. I adapted the char2subword module to multilingual BERT using pre-training from English, Spanish, Hindi, Arabic, and Nepali text. This step allows the module to be contextualized with the masked language modeling objective while preserving parameters for multilingual BERT fixed. Additionally, this strategy allows the model to be adapted to different domains and languages, not only social media and code-switching.

3. Releasing a self-contained char2subword module that can be used as a drop-in alternative for researchers and NLP practitioners to explore the adaptability of the model to other settings.

9.4 Future Work

While the proposed methods proved useful, modeling noise from social media text is a challenging task that still requires more work.

The module leveraging phonological spelling captures well part of the noise in SM text. However, this method requires to convert the text from the original alphabet to the international phonetic alphabet (IPA). One problem with this approach is that the input language may not be known during down-stream inference. Therefore, applying an IPA converter can become a source of errors. One potential solution is to learn a phonological space during pre-training with multilingual BERT. Like the position and type embeddings, the model can have an additional embedding table that supports phonetics. This strategy still depends on

IPA mappings from the original scripts, but the advantage is that all the languages are known during pre-training. Other problems would remain, such as IPA conversion for transliterated code-switching text or languages not supported by the IPA converters.

In the case of dependency trees (Chapter 4), the performance of the model is highly dependent on the quality of the trees. As shown by the relative self-attention, having a Cartesian product, the relationships can improve the model's judgment by connecting isolated trees. A more flexible method is graph convolutional networks (GCN) where the adjacency matrix uses smoothing to mitigate the entirely isolated segments of the text. With the same goal, it is possible to explore further the Transformer architecture (either from a pre-trained model like BERT or not), where the multi-head self-attention also conceives a graph from its self-attention interactions.

The char2subword module is a robust method that can adapt to code-switching and handle spelling issues. Nevertheless, some drawbacks can be improved in this method. For instance, the robustness incorporated during the approximation phase can also introduce ambiguity. I showed an example of this scenario, where a transliterated Hindi word was predicted as English because of the ambiguity raised by a single letter (e.g., beech in Hindi vs. beach in English). More importantly, a substantial improvement in this work would be to fully pre-train a language model with character-level inputs. While character inputs can become unreasonably long sequences during pre-training (e.g., a sequence of characters that accounts for 512 words), I propose to keep characters in the scope of a word. This keeps representations at the word level across the Transformer layers, preserving semantics. Then, the masked language model (MLM) objective can be computed for characters after one-to-many decoding (alternatively, having both character and word level MLM objectives). The decoding takes the word representation and generates the original sequence of characters. This method's benefit is that the semantics are not lost because the model is not entirely

character level. Simultaneously, the loss is computed to penalize inflections and morphology instead of fully learning different word vectors that share many linguistic properties.

Bibliography

[1] Aguilar, G., AlGhamdi, F., Soto, V., Diab, M., Hirschberg, J., and Solorio, T. Named entity recognition on code-switched data: Overview of the CALCS 2018 shared task. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching* (Melbourne, Australia, July 2018), Association for Computational Linguistics, pp. 138–147.

[2] Aguilar, G., Kar, S., and Solorio, T. LinCE: A centralized benchmark for linguistic code-switching evaluation. In *Proceedings of the 12th Language Resources and Evaluation Conference* (Marseille, France, May 2020), European Language Resources Association, pp. 1803–1813.

[3] Aguilar, G., Ling, Y., Zhang, Y., Yao, B., Fan, X., and Guo, C. Knowledge distillation from internal representations. *Proceedings of the AAAI Conference on Artificial Intelligence 34* (Feb. 2020), 7350–7357.

[4] Aguilar, G., López-Monroy, A. P., González, F., and Solorio, T. Modeling noisiness to recognize named entities using multitask neural networks on social media. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (New Orleans, Louisiana, June 2018), Association for Computational Linguistics, pp. 1401–1412.

[5] Aguilar, G., Maharjan, S., López-Monroy, A. P., and Solorio, T. A multi-task approach for named entity recognition in social media data. In *Proceedings of the 3rd Workshop on Noisy User-generated Text* (Copenhagen, Denmark, Sept. 2017), Association for Computational Linguistics, pp. 148–153.

[6] Aguilar, G., McCann, B., Niu, T., Rajani, N., Keskar, N., and Solorio, T. Char2subword: Extending the subword embedding space from pre-trained models using robust character compositionality. *arXiv preprint arXiv:2010.12730* (2020).

[7] Aguilar, G., Rozgic, V., Wang, W., and Wang, C. Multimodal and multi-view models for emotion recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (Florence, Italy, July 2019), Association for Computational Linguistics, pp. 991–1002.

[8] Aguilar, G., and Solorio, T. Dependency-aware named entity recognition with relative and global attentions. *arXiv preprint arXiv:1909.05166* (2019).

[9] Aguilar, G., and Solorio, T. From English to code-switching: Transfer learning with strong morphological clues. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (Online, July 2020), Association for Computational Linguistics, pp. 8033–8044.

[10] Akbik, A., Bergmann, T., and Vollgraf, R. Pooled contextualized embeddings for named entity recognition. 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)* (Minneapolis, Minnesota, June 2019), Association for Computational Linguistics, pp. 724–728.

[11] Akbik, A., Blythe, D., and Vollgraf, R. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics* (Santa Fe, New Mexico, USA, Aug. 2018), Association for Computational Linguistics, pp. 1638–1649.

[12] Al-Badrashiny, M., and Diab, M. LILI: A simple language independent approach for language identification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers* (Osaka, Japan, Dec. 2016), The COLING 2016 Organizing Committee, pp. 1211–1219.

[13] AlGhamdi, F., Molina, G., Diab, M., Solorio, T., Hawwari, A., Soto, V., and Hirschberg, J. Part of speech tagging for code switched data. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching* (Austin, Texas, Nov. 2016), Association for Computational Linguistics, pp. 98–107.

[14] Augenstein, I., Derczynski, L., and Bontcheva, K. Generalisation in named entity recognition: A quantitative analysis. *Computer Speech & Language 44* (01 2017).

[15] Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[16] Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (2015).

[17] Baldwin, T., de Marneffe, M.-C., Han, B., Kim, Y.-B., Ritter, A., and Xu, W. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text* (Beijing, China, July 2015), Association for Computational Linguistics, pp. 126–135.

[18] Bali, K., Sharma, J., Choudhury, M., and Vyas, Y. "I am borrowing ya mixing ?" an analysis of English-Hindi code mixing in Facebook. In *Proceedings of the First Workshop on Computational Approaches to Code Switching* (Doha, Qatar, Oct. 2014), Association for Computational Linguistics, pp. 116–126.

[19] Bender, O., Och, F. J., and Ney, H. Maximum entropy models for named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003* (2003), W. Daelemans and M. Osborne, Eds., pp. 148–151.

[20] Bharadwaj, A., Mortensen, D., Dyer, C., and Carbonell, J. Phonologically aware neural model for named entity recognition in low resource transfer settings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (Austin, Texas, November 2016), Association for Computational Linguistics, pp. 1462–1472.

[21] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics 5* (2017), 135–146.

[22] Bontcheva, K., Cunningham, H., Tablan, V., Maynard, D., and Hamza, O. Using gate as an environment for teaching nlp. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1* (Stroudsburg, PA, USA, 2002), ETMTNLP '02, Association for Computational Linguistics, pp. 54–62.

[23] Borthwick, A., Sterling, J., Agichtein, E., and Grishman, R. Nyu: Description of the mene named entity system as used in muc-7. In *In Proceedings of the Seventh Message Understanding Conference (MUC-7)* (1998).

[24] Chandu, K., Loginova, E., Gupta, V., Genabith, J. v., Neumann, G., Chinnakotla, M., Nyberg, E., and Black, A. W. Code-mixed question answering challenge: Crowd-sourcing data and techniques. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching* (Melbourne, Australia, July 2018), Association for Computational Linguistics, pp. 29–38.

[25] Chen, S., Aguilar, G., Neves, L., and Solorio, T. A caption is worth a thousand images: Investigating image captions for multimodal named entity recognition, 2020.

[26] Chieu, H. L., and Ng, H. T. Named entity recognition with a maximum entropy approach. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003* (2003), pp. 160–163.

[27] Chinchor, N., and Robinson, P. Muc-7 named entity task definition. In *Proceedings of the 7th Conference on Message Understanding* (1997), vol. 29, pp. 1–21.

[28] Chiu, J. P., and Nichols, E. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics 4* (2016), 357–370.

[29] Choudhury, M., Chittaranjan, G., Gupta, P., and Das, A. Overview of FIRE 2014 Track on Transliterated Search. *Proceedings of FIRE* (2014), 68–89.

[30] Clark, K., Khandelwal, U., Levy, O., and Manning, C. D. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP* (Florence, Italy, Aug. 2019), Association for Computational Linguistics, pp. 276–286.

[31] Curran, J., and Clark, S. Language independent NER using a maximum entropy tagger. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003* (2003), pp. 164–167.

[32] Das, A. Tool contest on POS tagging for code-mixed Indian social media (Facebook, Twitter, and Whatsapp) text. In *Proceedings of the 13th International Conference on Natural Language Processing* (2016). retrieved 05-10-2019.

[33] De Lannoy, G., François, D., Delbeke, J., and Verleysen, M. Weighted conditional random fields for supervised interpatient heartbeat classification. *IEEE Transactions on Biomedical Engineering 59*, 1 (2011), 241–247.

[34] de Marneffe, M.-C., Dozat, T., Silveira, N., Haverinen, K., Ginter, F., Nivre, J., and Manning, C. D. Universal Stanford dependencies: A cross-linguistic typology. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)* (Reykjavik, Iceland, May 2014), European Languages Resources Association (ELRA), pp. 4585–4592.

[35] Derczynski, L., Maynard, D., Rizzo, G., van Erp, M., Gorrell, G., Troncy, R., Petrak, J., and Bontcheva, K. Analysis of named entity recognition and linking for tweets. *Information Processing & Management 51*, 2 (2015), 32 – 49.

[36] Derczynski, L., Nichols, E., van Erp, M., and Limsopatham, N. Results of the WNUT2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text* (Copenhagen, Denmark, Sept. 2017), Association for Computational Linguistics, pp. 140–147.

[37] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. 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)* (Minneapolis, Minnesota, June 2019), Association for Computational Linguistics, pp. 4171–4186.

[38] Dror, R., Baumer, G., Shlomov, S., and Reichart, R. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Melbourne, Australia, July 2018), Association for Computational Linguistics, pp. 1383–1392.

[39] Dyer, C., Ballesteros, M., Ling, W., Matthews, A., and Smith, N. A. Transition-based dependeny parsing with stack long short-term memory. In *Proceedings of the* 53^{rd} *Annual Meeting of the Association of Computational Linguistics and the* 7^{th} *International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP 2015)* (2015), ACL.

[40] Ferguson, C. A. Diglossia. *Word 15*, 2 (1959), 325–340.

[41] Finkel, J. R., Grenager, T., and Manning, C. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics* (Stroudsburg, PA, USA, 2005), ACL '05, Association for Computational Linguistics, pp. 363–370.

[42] Florian, R., Ittycheriah, A., Jing, H., and Zhang, T. Named entity recognition through classifier combination. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003* (2003), pp. 168–171.

[43] Gal, Y., and Ghahramani, Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. *arXiv:1506.02142* (2015).

[44] Gambäck, B., and Das, A. On measuring the complexity of code-mixing. In *Proceedings of the 11th International Conference on Natural Language Processing, Goa, India* (2014), pp. 1–7.

[45] Geetha, P., Chandu, K., and Black, A. W. Tackling code-switched NER: Participation of CMU. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching* (Melbourne, Australia, July 2018), Association for Computational Linguistics, pp. 126–131.

[46] Godin, F., Vandersmissen, B., De Neve, W., and Van de Walle, R. Multimedia lab @ ACL WNUT NER shared task: Named entity recognition for twitter microposts using distributed word representations. In *Proceedings of the Workshop on Noisy User-generated Text* (Beijing, China, July 2015), Association for Computational Linguistics, pp. 146–153.

[47] Grishman, R., and Sundheim, B. Message understanding conference - 6: A brief history. In *COLING-1996 Vol. 1* (1996), pp. 466–471.

[48] Gururangan, S., Marasović, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., and Smith, N. A. Don't stop pretraining: Adapt language models to

domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (Online, July 2020), Association for Computational Linguistics, pp. 8342–8360.

[49] Hamed, I., Elmahdy, M., and Abdennadher, S. Collection and Analysis of Code-switch Egyptian Arabic-English Speech Corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)* (Miyazaki, Japan, May 7-12, 2018 2018), N. C. C. chair), K. Choukri, C. Cieri, T. Declerck, S. Goggi, K. Hasida, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, S. Piperidis, and T. Tokunaga, Eds., European Language Resources Association (ELRA).

[50] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 770–778.

[51] He, X., Haffari, G., and Norouzi, M. Dynamic programming encoding for subword segmentation in neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (Online, July 2020), Association for Computational Linguistics, pp. 3042–3051.

[52] Hendrycks, D., and Gimpel, K. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR abs/1606.08415* (2016).

[53] Hochreiter, S., and Schmidhuber, J. Long Short-Term Memory. *Neural computation 9*, 8 (Nov. 1997), 1735–1780.

[54] Howard, J., and Ruder, S. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Melbourne, Australia, July 2018), Association for Computational Linguistics, pp. 328–339.

[55] Huang, Z., Xu, W., and Yu, K. Bidirectional LSTM-CRF models for sequence tagging. *CoRR abs/1508.01991* (2015).

[56] Jamatia, A., Gambäck, B., and Das, A. Part-of-speech tagging for code-mixed English-Hindi twitter and Facebook chat messages. In *Proceedings of the International Conference Recent Advances in Natural Language Processing* (Hissar, Bulgaria, Sept. 2015), INCOMA Ltd. Shoumen, BULGARIA, pp. 239–248.

[57] Jie Yang, S. L., and Zhang, Y. Design challenges and misconceptions in neural sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics (CONLING)* (2018).

[58] Kar, S., Aguilar, G., Lapata, M., and Solorio, T. Multi-view story characterization from movie plot synopses and reviews. In *Proceedings of the 2020 Conference*

on Empirical Methods in Natural Language Processing (EMNLP) (Online, Nov. 2020), Association for Computational Linguistics, pp. 5629–5646.

[59] Keskar, N. S., McCann, B., Varshney, L. R., Xiong, C., and Socher, R. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858* (2019).

[60] Khandelwal, A., Swami, S., Akhtar, S. S., and Shrivastava, M. Humor detection in English-Hindi code-mixed social media content : Corpus and baseline system. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)* (Miyazaki, Japan, May 2018), European Languages Resources Association (ELRA).

[61] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. *CoRR abs/1412.6980* (2014).

[62] Klein, D., Smarr, J., Nguyen, H., and Manning, C. D. Named entity recognition with character-level models. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4* (Stroudsburg, PA, USA, 2003), CONLL '03, Association for Computational Linguistics, pp. 180–183.

[63] Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (San Diego, California, June 2016), Association for Computational Linguistics, pp. 260–270.

[64] Limsopatham, N., and Collier, N. Bidirectional lstm for named entity recognition in twitter messages. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)* (Osaka, Japan, December 2016), The COLING 2016 Organizing Committee, pp. 145–152.

[65] Liu, L., Shang, J., Xu, F. F., Ren, X., Gui, H., Peng, J., and Han, J. Empower sequence labeling with task-aware neural language model. *CoRR abs/1709.04109* (2017).

[66] Liu, X., Zhang, S., Wei, F., and Zhou, M. Recognizing named entities in tweets. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1* (Stroudsburg, PA, USA, 2011), HLT '11, Association for Computational Linguistics, pp. 359–367.

[67] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized BERT pretraining approach. *CoRR abs/1907.11692* (2019).

[68] Loshchilov, I., and Hutter, F. SGDR: stochastic gradient descent with restarts. *CoRR abs/1608.03983* (2016).

[69] Luong, T., Pham, H., and Manning, C. D. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (Lisbon, Portugal, Sept. 2015), Association for Computational Linguistics, pp. 1412–1421.

[70] Ma, X., and Hovy, E. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Berlin, Germany, Aug. 2016), Association for Computational Linguistics, pp. 1064–1074.

[71] Mager, M., Çetinoğlu, Ö., and Kann, K. Subword-level language identification for intra-word code-switching. *Proceedings of the 2019 Conference of the North* (2019).

[72] Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., and McClosky, D. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (Baltimore, Maryland, June 2014), Association for Computational Linguistics, pp. 55–60.

[73] Mave, D., Maharjan, S., and Solorio, T. Language identification and analysis of code-switched social media text. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching* (Melbourne, Australia, July 2018), Association for Computational Linguistics, pp. 51–61.

[74] Mayfield, J., McNamee, P., and Piatko, C. Named entity recognition using hundreds of thousands of features. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4* (Stroudsburg, PA, USA, 2003), CONLL '03, Association for Computational Linguistics, pp. 184–187.

[75] McCallum, A., and Li, W. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4* (Stroudsburg, PA, USA, 2003), CONLL '03, Association for Computational Linguistics, pp. 188–191.

[76] Mikheev, A., Grover, C., and Moens, M. Description of the ltg system used for muc-7. In *In Proceedings of 7th Message Understanding Conference (MUC-7* (1998).

[77] Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *CoRR abs/1301.3781* (2013).

[78] Molina, G., AlGhamdi, F., Ghoneim, M., Hawwari, A., Rey-Villamizar, N., Diab, M., and Solorio, T. Overview for the second shared task on language

identification in code-switched data. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching* (Austin, Texas, Nov. 2016), Association for Computational Linguistics, pp. 40–49.

[79] Mortensen, D. R., Littell, P., Bharadwaj, A., Goyal, K., Dyer, C., and Levin, L. Panphon: A resource for mapping ipa segments to articulatory feature vectors. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers* (Osaka, Japan, December 2016), The COLING 2016 Organizing Committee, pp. 3475–3484.

[80] Nothman, J., Ringland, N., Radford, W., Murphy, T., and Curran, J. R. Learning multilingual named entity recognition from wikipedia. *Artif. Intell. 194* (Jan. 2013), 151–175.

[81] Owoputi, O., O'Connor, B., Dyer, C., Gimpel, K., Schneider, N., and Smith, N. A. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (Atlanta, Georgia, June 2013), Association for Computational Linguistics, pp. 380–390.

[82] Partanen, N., Lim, K., Riessler, M., and Poibeau, T. Dependency parsing of code-switching data with cross-lingual feature representations. In *Proceedings of the Fourth International Workshop on Computational Linguistics of Uralic Languages* (Helsinki, Finland, Jan. 2018), Association for Computational Linguistics, pp. 1–17.

[83] Patwa, P., Aguilar, G., Kar, S., Pandey, S., PYKL, S., Garrette, D., Gambäck, B., Chakraborty, T., Solorio, T., and Das, A. SemEval-2020 Task 9: Overview of Sentiment Analysis of Code-Mixed Tweets. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2020)* (Barcelona, Spain, Sept. 2020), Association for Computational Linguistics.

[84] Peng, N., and Dredze, M. Improving named entity recognition for chinese social media with word segmentation representation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (Berlin, Germany, August 2016), Association for Computational Linguistics, pp. 149–155.

[85] Peng, N., and Dredze, M. Improving named entity recognition for Chinese social media with word segmentation representation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (Berlin, Germany, Aug. 2016), Association for Computational Linguistics, pp. 149–155.

[86] Pennington, J., Socher, R., and Manning, C. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural*

Language Processing (EMNLP) (Doha, Qatar, Oct. 2014), Association for Computational Linguistics, pp. 1532–1543.

[87] Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (New Orleans, Louisiana, June 2018), Association for Computational Linguistics, pp. 2227–2237.

[88] Peters, M., Neumann, M., Zettlemoyer, L., and Yih, W.-t. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (Brussels, Belgium, October-November 2018), Association for Computational Linguistics, pp. 1499–1509.

[89] Petrov, S., Das, D., and McDonald, R. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)* (Istanbul, Turkey, May 2012), European Languages Resources Association (ELRA), pp. 2089–2096.

[90] Pierre, J. M. Mining knowledge from text collections using automatically generated metadata. In *Proceedings of the 4th International Conference on Practical Aspects of Knowledge Management* (London, UK, UK, 2002), PAKM '02, Springer-Verlag, pp. 537–548.

[91] Pratapa, A., Choudhury, M., and Sitaram, S. Word embeddings for code-mixed language processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (Brussels, Belgium, Oct.-Nov. 2018), Association for Computational Linguistics, pp. 3067–3072.

[92] Raghavi, K. C., Chinnakotla, M. K., and Shrivastava, M. Answer ka type kya he?: Learning to classify questions in code-mixed language. In *Proceedings of the 24th International Conference on World Wide Web* (2015), ACM, pp. 853–858.

[93] Recasens, M., Màrquez, L., Sapena, E., Martí, M. A., Taulé, M., Hoste, V., Poesio, M., and Versley, Y. Semeval-2010 task 1: Coreference resolution in multiple languages. In *Proceedings of the 5th International Workshop on Semantic Evaluation* (2010), Association for Computational Linguistics, pp. 1–8.

[94] Ritter, A., Clark, S., Mausam, and Etzioni, O. Named entity recognition in tweets: An experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (Stroudsburg, PA, USA, 2011), EMNLP '11, Association for Computational Linguistics, pp. 1524–1534.

[95] Rothman, J., and Rell, A. B. A linguistic analysis of spanglish: Relating language to identity. *Linguistics and the Human Sciences 1*, 3 (2007), 515–536.

[96] Roy, R. S., Choudhury, M., Majumder, P., and Agarwal, K. Overview of the FIRE 2013 Track on Transliterated Search. In *Post-Proceedings of the 4th and 5th Workshops of the Forum for Information Retrieval Evaluation* (2013), ACM, p. 4.

[97] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Nature 323*, 6088 (1986), 533–536.

[98] Samih, Y., Maharjan, S., Attia, M., Kallmeyer, L., and Solorio, T. Multilingual code-switching identification via LSTM recurrent neural networks. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching* (Austin, Texas, Nov. 2016), Association for Computational Linguistics, pp. 50–59.

[99] Sechidis, K., Tsoumakas, G., and Vlahavas, I. On the stratification of Multi-label Data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (2011), Springer, pp. 145–158.

[100] Sennrich, R., Haddow, B., and Birch, A. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Berlin, Germany, Aug. 2016), Association for Computational Linguistics, pp. 1715–1725.

[101] Sequiera, R., Choudhury, M., and Bali, K. POS Tagging of Hindi-English Code Mixed Text from Social Media: Some Machine Learning Experiments. In *2015 Proceedings of International Conference on NLP* (December 2015), NLPAI.

[102] Sequiera, R., Choudhury, M., Gupta, P., Rosso, P., Kumar, S., Banerjee, S., Naskar, S. K., Bandyopadhyay, S., Chittaranjan, G., Das, A., et al. Overview of FIRE-2015 Shared Task on Mixed Script Information Retrieval. In *FIRE Workshops* (2015), vol. 1587, pp. 19–25.

[103] Singh, K., Sen, I., and Kumaraguru, P. Language identification and named entity recognition in Hinglish code mixed tweets. In *Proceedings of ACL 2018, Student Research Workshop* (Melbourne, Australia, July 2018), Association for Computational Linguistics, pp. 52–58.

[104] Singh, K., Sen, I., and Kumaraguru, P. A twitter corpus for Hindi-English code mixed POS tagging. In *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media* (Melbourne, Australia, July 2018), Association for Computational Linguistics, pp. 12–17.

[105] Sitaram, S., Chandu, K. R., Rallabandi, S. K., and Black, A. W. A survey of code-switched speech and language processing. *arXiv preprint arXiv:1904.00784* (2019).

[106] Solorio, T., Blair, E., Maharjan, S., Bethard, S., Diab, M., Ghoneim, M., Hawwari, A., AlGhamdi, F., Hirschberg, J., Chang, A., and Fung, P.

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* (Doha, Qatar, Oct. 2014), Association for Computational Linguistics, pp. 62–72.

[107] Solorio, T., and Liu, Y. Part-of-speech tagging for english-spanish code-switched text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (Stroudsburg, PA, USA, 2008), EMNLP '08, Association for Computational Linguistics, pp. 1051–1060.

[108] Soto, V., and Hirschberg, J. Crowdsourcing universal part-of-speech tags for code-switching. *Interspeech 2017* (Aug 2017).

[109] Soto, V., and Hirschberg, J. Joint part-of-speech and language ID tagging for code-switched data. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching* (Melbourne, Australia, July 2018), Association for Computational Linguistics, pp. 1–10.

[110] Strauss, B., Toma, B., Ritter, A., de Marneffe, M.-C., and Xu, W. Results of the WNUT16 named entity recognition shared task. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)* (Osaka, Japan, Dec. 2016), The COLING 2016 Organizing Committee, pp. 138–144.

[111] Sutskever, I., Martens, J., Dahl, G., and Hinton, G. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning* (Atlanta, Georgia, USA, 17–19 Jun 2013), S. Dasgupta and D. McAllester, Eds., vol. 28 of *Proceedings of Machine Learning Research*, PMLR, pp. 1139–1147.

[112] Tai, K. S., Socher, R., and Manning, C. D. 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)* (Beijing, China, July 2015), Association for Computational Linguistics, pp. 1556–1566.

[113] Tjong Kim Sang, E. F., and De Meulder, F. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003* (2003), pp. 142–147.

[114] Tristram, H. L. C. *How Celtic is Standard English?* Nauka, St Petersburg, Russia, 1999.

[115] Trivedi, S., Rangwani, H., and Kumar Singh, A. IIT (BHU) submission for the ACL shared task on named entity recognition on code-switched data. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*

(Melbourne, Australia, July 2018), Association for Computational Linguistics, pp. 148–153.

[116] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems* (2017), Curran Associates, Inc., pp. 5998–6008.

[117] Vilares, D., Alonso, M. A., and Gómez-Rodríguez, C. Sentiment analysis on monolingual, multilingual and code-switching twitter corpora. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis* (2015), Association for Computational Linguistics, pp. 2–8.

[118] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP* (Brussels, Belgium, Nov. 2018), Association for Computational Linguistics, pp. 353–355.

[119] Wang, C., Cho, K., and Kiela, D. Code-switched named entity recognition with embedding attention. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching* (Melbourne, Australia, July 2018), Association for Computational Linguistics, pp. 154–158.

[120] Whitelaw, C., and Patrick, J. Named entity recognition using a character-based probabilistic approach. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4* (Stroudsburg, PA, USA, 2003), CONLL '03, Association for Computational Linguistics, pp. 196–199.

[121] Winata, G. I., Lin, Z., and Fung, P. Learning multilingual meta-embeddings for code-switching named entity recognition. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)* (Florence, Italy, Aug. 2019), Association for Computational Linguistics, pp. 181–186.

[122] Winata, G. I., Wu, C.-S., Madotto, A., and Fung, P. Bilingual character representation for efficiently addressing out-of-vocabulary words in code-switching named entity recognition. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching* (Melbourne, Australia, July 2018), Association for Computational Linguistics, pp. 110–114.

[123] Yu, J., Bohnet, B., and Poesio, M. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (Online, July 2020), Association for Computational Linguistics, pp. 6470–6476.

[124] Zhang, K. W., and Bowman, S. R. Language modeling teaches you more syntax than translation does: Lessons learned through auxiliary task analysis. *EMNLP 2018* (2018), 359.