Dotless Arabic Text for Natural Language Processing

Maged S. Al-Shaibani[1] and Irfan Ahmad[1,2]

[1]King Fahd University of Petroleum and Minerals Information & Computer Science Department
g201381710@kfupm.edu.sa

[2]SDAIA-KFUPM Joint Research Center for AI
irfan.ahmad@kfupm.edu.sa

This article introduces a novel representation of Arabic text as an alternative approach for Arabic NLP, inspired by the dotless script of ancient Arabic. We explored this representation through extensive analysis on various text corpora, differing in size and domain, and tokenized using multiple tokenization techniques. Furthermore, we examined the information density of this representation and compared it with the standard dotted Arabic text using text entropy analysis. Utilizing parallel corpora, we also drew comparisons between Arabic and English text analysis to gain additional insights. Our investigation extended to various upstream and downstream NLP tasks, including language modeling, text classification, sequence labeling, and machine translation, examining the implications of both the representations. Specifically, we performed seven different downstream tasks using various tokenization schemes comparing the standard dotted text with dotless Arabic text representations. Performance using both the representations was comparable across different tokenizations. However, dotless representation achieves these results with significant reduction in vocabulary sizes, and in some scenarios showing reduction of up to 50%. Additionally, we present a system that restores dots to the dotless Arabic text. This system is useful for tasks that require Arabic texts as output.

1. Introduction

Arabic serves as the official language across 22 nations within the Arab world, with a native speaker base exceeding 400 million speakers (Guellil et al. 2021). Moreover, it is a secondary language for a large community of non-Arabic Muslims. Recognized as one of the United Nation's six official languages (Boudad et al. 2018), Arabic occupies a significant linguistic sphere. Notably, it has ascended to become the fourth most

Action Editor: Preslav Nakov. Submission received: 26 December 2023; revised version received: 11 May 2024; accepted for publication: 24 July 2024.

https://doi.org/10.1162/coli_a_00535

© 2024 Association for Computational Linguistics
Published under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0) license

prevalent language on the Internet, experiencing rapid growth in user engagement between 2013 and 2018 (Boudad et al. 2018).

Arabic can be categorized into three categories. Classical Arabic, which can be found in sacred texts, tightly connected to the Islamic cultural and religious heritage, as well as ancient Arabic literature. Modern Standard Arabic (MSA), which is the formal language used in official documentation and news. Finally, Dialectical or Colloquial Arabic, which encompasses a multitude of regionally specific vernaculars derived from MSA, utilized extensively in day-to-day casual communication.

Arabic, similar to other Semitic languages, is distinguished for its dense morphology, characterized by a non-concatenative structure. In this morphological structure, words are crafted by interweaving vowel letters amidst the root consonants, coupled with the incorporation of prefixes and affixes. This intricate morphology poses formidable challenges for prevailing natural language processing (NLP) tools. For instance, the expansion of the language's lexicon surpasses that of languages such as English. Research by Alotaiby, Alkharashi, and Foda (2009) underscores this, revealing that the vocabulary size of an Arabic corpus is twice that of an English corpus where both were drawn from a similar domain, despite containing a roughly equivalent number of tokens.

The Arabic script is a right-to-left cursive writing system. Each letter exhibits diverse glyphs based on its placement within a word. There are a total of 28 letters; 25 of these represent consonants. Notably, nearly half of these letters share the same base glyphs, known as *rasm*, but are differentiated by the absence or presence of dots either above or below their *rasm*, as illustrated in Figure 1. Additionally, the script incorporates 8 supplementary symbols known as diacritics. These diacritics serve the purpose of short vowels offering phonetic guidance, thereby mitigating potential ambiguities.

During the initial stages of the language writing development, the utilization of dots lacked prominence and formalization (Bentouati 2020). This stemmed from the limited popularity of writing practices during that era, coupled with the relative ease with which native speakers could disambiguate dotless text from the context. The initial trials of standardized dots usage along with *rasms* is attributed to Abu Al-Aswad Al-Dualli, a renowned Arabic scholar, who introduced dots in the sacred text of the Holy Quran. In this stage, dots functioned similarly to diacritics representing the word's grammatical state within a sentence. However, this practice was subsequently superseded by the utilization of dots to differentiate letters sharing identical *rasm* and diacritics to help in phonetic guidance and represent the word's grammatical state.

The primary motivation behind this formalization was to facilitate Arabic comprehension for non-native speakers, thereby safeguarding against foreign influences on subsequent generations of native speakers. Figure 2 showcases an Arabic sentence depicted in three distinct formats. The first is the dotless sentence, followed by the

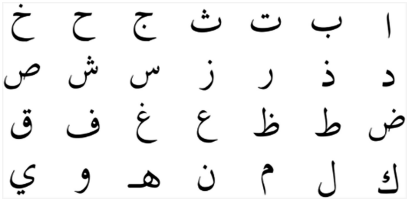


Figure 1
Characters in the Arabic script.

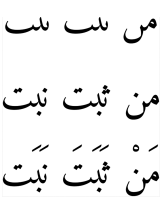


Figure 2
Arabic text with and without dots and diacritics.

sentence with dots, and finally, the sentence incorporating both dots and diacritics. Notably, the second word illustrates a potential ambiguity for multiple meanings, where this can be resolved through the use of dots.

Utilizing dotless Arabic text presents a promising avenue to address various challenges encountered in Arabic NLP. The inherent density of Arabic morphology often results in a considerably large vocabulary. However, dotless text could mitigate this by mapping many dotted words into a single dotless homographic word. Moreover, using dotless text can be useful in the area of automatic recognition of ancient parchments where the script used was predominantly dotless. Furthermore, the findings in this study can be extended to augmenting processing tools for social media platforms, especially with the rise of a recent trend encouraging users to use dotless text instead of dotted text in posts that contravene platform regulations (Rom and Bar 2021). These studies have highlighted that despite the absence of dots, native speakers retain a substantial ability to recognize and interpret dotless text.

While native Arabic speakers can understand text without dots, this research aims to explore whether this capability can be extended to Arabic NLP. Our primary interest is to evaluate the effectiveness of context-aware deep learning models when applied to dotless Arabic text, comparing their performance to that of standard dotted Arabic text. To investigate this, we assessed both text representations across various NLP tasks, including language modeling, text classification, sequence labeling, and machine translation. We hypothesize that by leveraging sufficient contextual information, these advanced algorithms may achieve a level of proficiency in interpreting dotless text that approaches human-level comprehension.

Before getting into an experimental analysis to evaluate the applicability and utility of dotless Arabic text as a representation for NLP tasks, conducting a preliminary statistical analysis would offer valuable insights. This statistical examination encompasses token counts across multiple levels of tokenization granularity, namely, words, subwords, and characters. This analysis also extends to cover quantitative analytical tools such as Zipf's and Heap's laws.

This study tries to bridge the gap in Arabic NLP, introducing the following contributions. First, it introduces a novel method for representing Arabic text by removing dots from letters, providing a new perspective on text representation. It also offers a detailed comparative analysis of dotted and dotless Arabic text across five varied text corpora, utilizing multiple tokenization levels (character, word, and subword) to investigate the approach generalization. The study extends further by comparing these representations with English text, thus broadening the research's scope and depth. Moreover, the paper conducts an in-depth examination of language modeling techniques, including statistical n-grams and neural models, applied to both dotted and dotless text from diverse domains. The article further explores the impact of dotless representation on

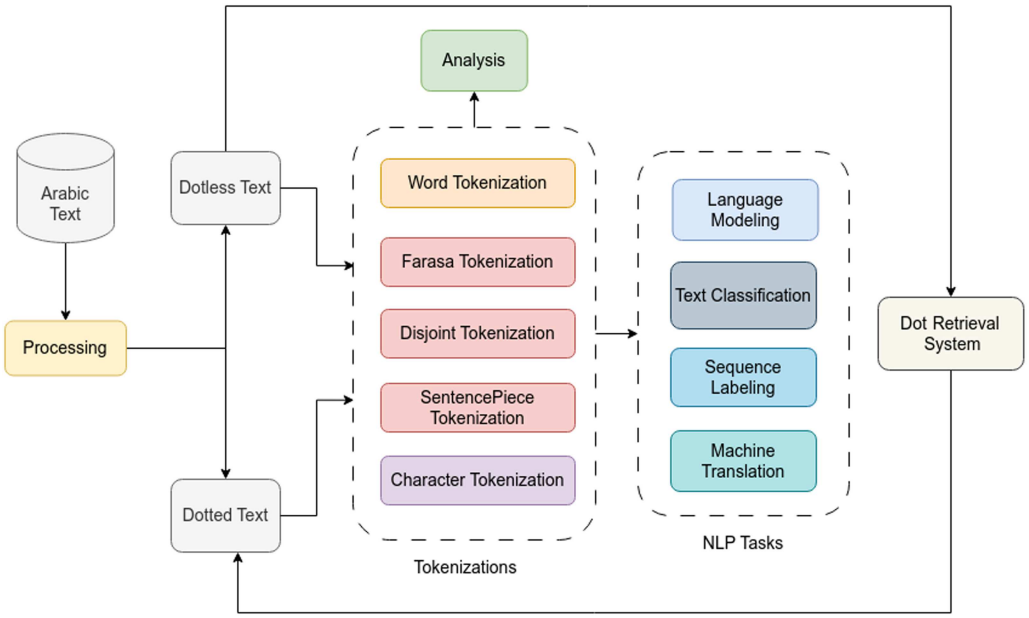


Figure 3
Paper methodology and main contributions.

various downstream tasks covering text classification, sequence labeling, and machine translation. Additionally, this article presents a system that is capable of restoring dots to dotless text, thereby offering practical utility in text processing. Figure 3 illustrates the main contributions, which can be summarized as follows:

- A novel method to represent Arabic text for NLP by removing dots from the Arabic letters.
- A comprehensive analysis of Arabic text rigorously comparing dotted and dotless text conducted on five different corpora, varying in size and domains. The analysis encompasses multiple tokenization levels, specifically, character, word, and subword. Additionally, this study supplements its findings by conducting a comparative analysis between Arabic dotted and dotless text with English text, enhancing the depth and breadth of the research insights.
- An extensive examination and comparison of dotted and dotless text is conducted with two primary language modeling techniques—statistical *n*-grams and neural language models—utilizing corpora sourced from diverse domains.
- A system that restores dots to dotless Arabic text.
- An extensive evaluation and analysis of dotted and dotless text representation performance on various downstream NLP tasks. The tasks cover text classification, sequence labeling, and machine translation. We

conducted this analysis on various tokenization approaches as in the case of language modeling.

The remaining sections of this article are organized as follows: Section 2 surveys the advancements and developments relevant to this topic in the existing literature. Section 3 provides an overview of the corpora utilized in this study, detailing the preprocessing steps undertaken and elucidating the utilized tokenization methods. In Section 4, we present the various text analyses conducted to compare dotted and dotless text. Section 5 outlines our experiments involving language modeling. Section 6 presents our experiments on restoring dots to dotless text. Section 7 presents our experiments conducted on various downstream tasks. Finally, Section 8 concludes this study, summarizing the findings and presenting future directions for research.

2. Related Work

To the best of our knowledge, this is the first work that analyzes dotless Arabic text in the domain of natural language processing tasks. However, the utilization of dotless text in social media to circumvent content filtering algorithms has attracted recent attention. In response to this trend, Rom and Bar (2021) delved into methodologies to accommodate undotted text within the prevailing settings of language models trained predominantly on dotted text, particularly in the context of social media platforms. Their exploration yielded two primary approaches. The first approach involved reconfiguring the tokenizer either by removing dots from its tokens or expanding it with undotted variants of these tokens. These undotted versions were then mapped to the same identifier as their dotted counterparts, effectively enabling the language model to accept both dotted and dotless text. The second approach proposed a dotting algorithm aimed at restoring dots to the undotted text. Through this method, they demonstrated that the restored dotting approach produced results closely aligned with the original experiments conducted on dotted text when evaluated across various downstream tasks.

Alhathloul and Ahmad (2022) presented a deep learning-based approach aimed at restoring dots to dotless text. Using an extensive experimentation strategy across various datasets, they conducted an in-depth error analysis to assess the efficacy of their proposed method. Their approach relied on a Gated Recurrent Networks (GRUs) based sequence-tagging model, designed to generate dotted text as character sequence from dotless text as input sequence at the character level. In order to facilitate a fair comparison and evaluation, the researchers introduced a new metric called "dottization error rate." Across their experimentation involving four distinct datasets, their method exhibited promising results, achieving character error rates ranging between 2% and 5.5%, along with dottization error rates spanning from 4.2% to 11%.

Recent research has explored novel approaches to text representation in English for natural language processing tasks. Al-Shaibani and Ahmad (2023) proposed a consonant-based representation of English text, introducing two variants: a "consonants-only" approach where vowels are entirely removed, and a "masked-vowels" approach where vowels are replaced with a special symbol. Their work demonstrated that these representations can achieve comparable performance to standard text across various NLP tasks while significantly reducing vocabulary size and model parameters. Similarly, Alajrami, Margatina, and Aletras (2023) conducted an extensive

study on the role of input token characters in language models. They pre-trained models using small subsets of characters from individual tokens and found that even under extreme settings, such as using only the first character of each token, models retained a high percentage of performance compared to full-token models in standard NLU benchmarks. Both studies highlight the potential for more efficient text representations in NLP, suggesting that traditional approaches may contain redundant information and that simplified representations can maintain effectiveness while reducing computational demands.

Zipf's and Heap's laws serve as pivotal statistical tools in linguistics that link the number of tokens in a corpora to vocabulary size, offering profound insights when applied to language corpora, extensively examined in languages like English, Chinese, Norwegian, and Old German (Li 1992; Moreno-Sánchez, Font-Clos, and Corral 2016; Sicilia-Garcia et al. 2003; Welsh 1988). However, their application to Arabic remains relatively limited, potentially influenced by Arabic's perception as a low-resource language (Magueresse, Carles, and Heetderks 2020). Within Arabic language studies, two primary research objectives were studied. Firstly, analytical comparisons of Arabic with other languages have been explored (Alotaiby, Alkharashi, and Foda 2009; Alotaiby, Foda, and Alkharashi 2014; Al-Kadi 1998). Secondly, these analyses have been utilized to assess the quality of proposed corpora (Alarifi et al. 2012; Almeman and Lee 2013; Selab and Guessoum 2015; Khreisat 2009). Notably, much of this research has focused on evaluating extensive corpora within a single domain, predominantly newswire. Consequently, a notable research gap exists in conducting a comprehensive analysis encompassing diverse tokenization levels, linguistic units, and datasets spanning various domains within Arabic language studies.

Language models can be defined as mathematical models designed to predict the most probable word within a sequence of preceding words, representing a critical component in numerous natural language processing domains. Their utility extends across various tasks, including generative applications like machine translation (Diab, Ghoneim, and Habash 2007), optical character recognition enhancement (Smith 2011), and the augmentation of automatic speech recognition systems (Abushariah et al. 2010). Leveraging their capacity to encapsulate implicit linguistic information, neural language models find applications in sequence labeling tasks such as part-of-speech tagging and named entity recognition through a transfer learning style. Within these contexts, language models play a pivotal role in generating high-quality embeddings to enhance subsequent classification processes.

The evolution of language modeling boasts a substantial developmental history. Initially, these models sought to learn the joint probabilities of sequences within expansive training corpora. Using the Markov principle, these sequences were pruned to encompass orders 2, 3, 4, or higher n tokens, commonly known as n-grams, aiming to achieve an acceptable level of generalization. However, this approach suffered from an inherent challenge of dealing with Out-Of-Vocabulary (OOV) tokens, wherein the model encountered difficulty computing the probability of words absent from the training corpus. To address this challenge, various smoothing techniques have been proposed, including Laplace (additive) smoothing, Katz smoothing (Katz 1987), and Kneser-Ney smoothing (Kneser and Ney 1995). Subsequently, specialized toolkits emerged as standardized solutions for constructing these language models using sufficiently extensive corpora. Prominent examples of such toolkits include KenLM (Heafield 2011) and SriLM (Stolcke 2002). These toolkits have streamlined the development process by providing efficient means to build robust language models, marking significant strides in the field of language modeling.

Neural language models surpassed statistical counterparts (Bengio, Ducharme, and Vincent 2000), with Recurrent Neural Networks (RNNs) initially dominating sequence modeling. However, they struggled with issues like vanishing or exploding gradients, particularly evident in lengthy sequences. To address this, Long Short-Term Memory (LSTM) architecture was introduced (Hochreiter and Schmidhuber 1997), albeit slower and more challenging to train. GRU (Chung et al. 2014) architecture has been introduced as a compromise between vanilla RNNs and LSTMs. Recently, transformer-based models (Vaswani et al. 2017) sparked a revolution in NLP, overcoming the long recurrence limitation of previous recurrent networks. Pretrained models like BERT (Devlin et al. 2019), Wav2Vec2 (Baevski et al. 2020), and GPT-3 (Brown et al. 2020) attained state-of-the-art performance across various NLP tasks using this architecture. Nonetheless, the attention mechanism, intrinsic to transformers, exhibited limitations in mastering simpler tasks easily tackled by earlier models (Dehghani et al. 2018; Chernyavskiy, Ilvovsky, and Nakov 2021).

3. Text Corpora, Preprocessing, and Tokenization

In this section, we introduce the corpora utilized in this research. We also present the preprocessing steps we applied for analysis and experimentations. We, furthermore, overview the tokenization methods we used across this research.

3.1 Text Corpora

We selected corpora from different domains with different sizes to evaluate dotless text as an alternative representation for Arabic NLP. The following are the selected corpora:

- **Quran** (Aloufi 2019): A dataset compiled from Quran verses. In Quran, most chapters, called Surah, start with a special sentence, called Basmala. In order not to confuse the training with this repetitive sentence, we removed it from the beginning of each chapter except the first one. The total number of samples is 6,236 sentences.
- **Sanadset dataset** (Mghari, Bouras, and El Hibaoui 2022): This dataset is a collection of sacred sayings by the prophet Muhammed, called Hadeeth. Each Hadeeth contains a Matn, which is the sacred text; and a Sanad, which is the chain of narrators. We selected only the Matn. This dataset is interesting because its text is coherent with a relatively small vocabulary size compared to its size in terms of the total running text.
- **Ashaar (Poems)** (Alyafeai and Al-Shaibani 2020): This is a poetry dataset collected from various online sources. Samples extracted from this dataset comprise a set of 6 verses. Throughout this research, we will refer to this dataset as "Ashaar" or "Poems" dataset.
- **Newswire** dataset (El-Khair 2016): This dataset is a massive collection of news collected from various newswires. We only selected one newspaper, Alittihad.

- **Wikipedia** (Foundation): A dataset collected from Wikipedia dumps of Arabic articles up to 20 October 2022.

Table 1 presents detailed statistics with respect to dotted and dotless word tokens for each dataset. In this table, N represents total tokens in the running text, that is, all words; V represents the vocabulary size; and V' represents the vocabulary size of the dotless text. As presented in the table, there is a noticeable reduction in the dotless vocabulary compared with the dotted vocabulary. The reduction ranges from 10.3% to up to 37.3%. It can also be observed that the Quran dataset is the smallest dataset among our series. The Sanadset dataset contains very coherent text with many repetitive phrases. This can be deduced from the ratio of its unique vocabulary to the size of its running text. The Poems dataset, in contrast to the Sanadset dataset, has a large vocabulary relative to its running text size. This can be used as a measure to highlight the richness of the poetry dataset, attributed to the inherent creativity associated with poetry. Furthermore, classical Arabic poetry adheres to strict metrical and rhyming constraints, which compel poets to seek words that harmonize with the poem's rhyme scheme. Regarding the news and Wikipedia datasets, they are large corpora capturing various aspects and structures of the language. However, the news dataset has a much narrower domain than Wikipedia as can be deduced from their vocabulary sized compared to their running text size. From this table, it can be observed that Ashaar is richer than Sanadset. The Wikipedia dataset is richer than the news dataset, indicating that it covers wider and more diverse topics than the news dataset.

From the above statistics, it is clear that the datasets we chose are of different sizes and can be grouped into three categories. From the size of the running text N, the Quran text is a significantly small corpus, Sanadset and Ashaar are medium-sized corpora, and Wikipedia and news are larger corpora. Further noteworthy information to highlight is that the number of dotted characters, i.e., the vocabulary size at the character level, is 31 and the number of dotless characters is 19, representing a reduction by 38.7%.

3.2 Text Pre-Processing and Undotting

In this subsection, we present the procedure we followed to pre-process and undot text. For pre-processing, we removed any non-Arabic characters including numeral characters and punctuation symbols. Diacritics are also omitted. Further, as the letter Hamza can appear adjoined with other characters, usually vowels, we removed it, keeping only the adjoined letter. However, if it is found alone, it is kept unchanged.

Table 1
Datasets statistics with words tokenization covering dotted and dotless vocabulary.

Dataset	N	V	V′	V′/V
Quran	77.8K	14,748	13,229	89.70%
Sanadset	28.9M	317,331	227,590	71.72%
Ashaar	34.9M	1,007,279	631,487	62.69%
Wikipedia	177.4M	1,811,244	1,345,853	74.31%
News	134.9M	892,583	654,982	73.38%
Aggregated	376.2M	2,739,172	1,865,126	68.09%

For undotting, the table presented in Figure 4 illustrates Arabic letters and their corresponding dotless representation, depending on its position in a given word. It presents the letter shape across all its possible positions. Due to the cursive nature of the Arabic script, we tried to mimic the same writing shape for a few of the letters to its closest dotless letter if it comes either at the beginning or the middle of the word. For the letters ن and ي, if they come either at the beginning or the middle of the word, they are mapped to the dotless version of ب. Similarly, for the letter ق, if it comes at the beginning or middle of the word, it is mapped to the dotless version of ف.

3.3 Tokenization

Tokenization involves segmenting text into smaller units like words, characters, or in-between units known as subwords. Notably, subword tokenization emerged as an interesting approach due to its adaptability, being either language-dependent or language-agnostic. This method integrates benefits from different ends of the spectrum, presenting a versatile and robust means to represent input text. Alyafeai et al. (2023)

Isolated Shape	Dotless Shape	Beginning Shape	Dotless Shape	Middle Shape	Dotless Shape	End shape	Dotless Shape
ا	ا					ـا	ـا
ب ت ث	ب	بـ تـ ثـ	ـ	ـبـ ـتـ ـثـ	ـ	ـب ـت ـث	ـب
ج ح خ	ح	جـ حـ خـ	حـ	ـجـ ـحـ ـخـ	حـ	ـج ـح ـخ	ـح
د ذ	د					ـد ـذ	ـد
ر ز	ر					ـر ـز	ـر
س ش	س	سـ شـ	سـ	ـسـ ـشـ	سـ	ـس ـش	ـس
ص ض	ص	صـ ضـ	صـ	ـصـ ـضـ	صـ	ـص ـض	ـص
ط ظ	ط	طـ ظـ	طـ	ـطـ ـظـ	طـ	ـط ـظ	ـط
ع غ	ع	عـ غـ	عـ	ـعـ ـغـ	عـ	ـع ـغ	ـع
ف	ف	فـ	فـ	ـفـ	فـ	ـف	ـف
ق	ق	قـ	قـ	ـقـ	قـ	ـق	ـق
ك	ك	كـ	كـ	ـكـ	كـ	ـك	ـك
ل	ل	لـ	لـ	ـلـ	لـ	ـل	ـل
م	م	مـ	مـ	ـمـ	مـ	ـم	ـم
ن	ن	نـ	نـ	ـنـ	نـ	ـن	ـن
ه	ه	هـ	هـ	ـهـ	هـ	ـه	ـه
و	و					ـو	ـو
ي	ى	يـ	نـ	ـيـ	نـ	ـي	ـى
ء	ء					ـء	ـء

Figure 4
Arabic dotted letters mapped to their dotless variants depending on their position in the word.

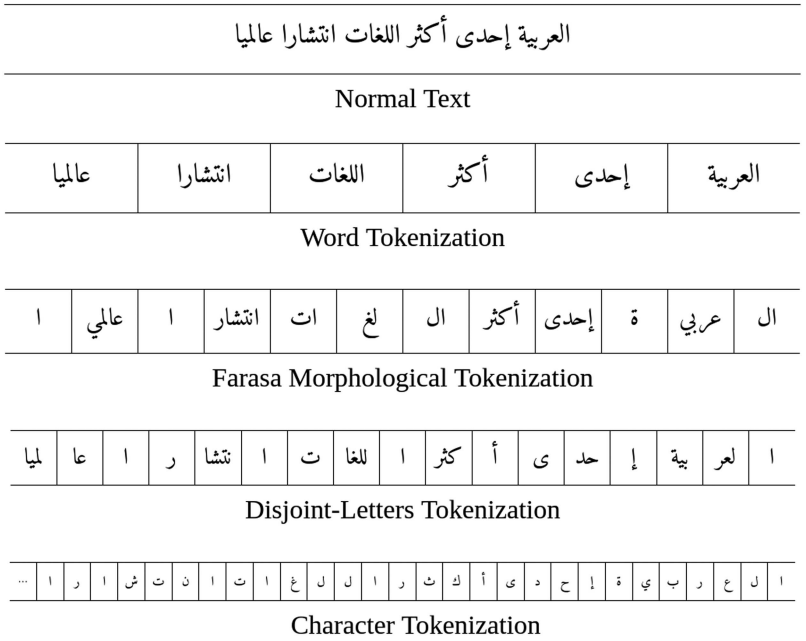


Figure 5
An Arabic sentence tokenized with different tokenization techniques.

conducted an in-depth comparison of various tokenization techniques for Arabic. They concluded that different tokenizations may be best suited to different tasks.

In this research, we studied the behavior of dotless text compared to dotted text using different tokenization techniques. We used word tokenization, character tokenization, morphology-based, and language-specific disjoint subword tokenizations in our analysis and across many NLP tasks. These tokenizations are described as follows.

- **Word tokenization**: This tokenization splits text into words.
- **Farasa morphological tokenization**: This tokenization splits text into morphemes. The tool used to perform this tokenization is called farasa (Abdelali et al. 2016), hence the name.
- **Disjoint-letters tokenization**: This tokenization is a language dependent tokenization introduced by Alyafeai et al. (2023). Although the Arabic script is cursive, some of its letters are written disconnected from their subsequent letter in the word. Based on this property, this tokenization splits the text into subwords. That is, each subword in this tokenization is a sequence of fully connected letters.
- **Character tokenization**: This tokenization splits text into characters.

Figure 5 shows an example of an Arabic sentence tokenized with each of the aforementioned tokenizations. To perform the tokenization task, we extended the

Table 2
Datasets statistics with farasa and disjoint-letters tokenizations covering dotted and dotless vocabulary.

Dataset	Farasa				Disjoint-Letters			
	N	V	V′	V′/V	N	V	V′	V′/V
Quran	129.8K	7,677	6,074	79.04%	164.6K	6,484	4,303	66.36%
Sanadset	44.1M	105,469	74,674	70.85%	55.9M	76,495	37,412	48.91%
Ashaar	57.4M	361,739	242,735	67.09%	72.4M	191,393	79,968	41.78%
Wikipedia	303.8M	1,002,148	791,084	78.91%	412.8M	266,822	118,150	44.28%
News	241.2M	344,627	263,143	76.36%	326.0M	151,557	68,943	45.49%
Aggregated	646.6M	1,417,437	1,061,481	74.92%	867.4M	402,019	162,389	40.39%

tokenizers package, named tkseem,[1] to fit our needs while utilizing their implemented functionalities.

4. Text Analysis

This section outlines the analysis we performed to compare dotless text with its dotted counterpart. It begins by providing a comprehensive overview of vocabulary statistics across various tokenization levels and datasets. Next, it examines the representation of both dotless and dotted texts by applying Zipf's and Heap's laws for linguistic analysis. Finally, the section ends with an entropy analysis of both the text representations.

4.1 Vocabulary Statistics

In Section 3.1, we presented the vocabulary analysis at the word level. Table 2 presents the statistics for the two subword tokenizations used in this study. As in the case of Table 1, N represents total tokens in the running text, V represents the vocabulary size for the standard Arabic text, and V′ represents the vocabulary size of the dotless text for a given tokenization. The table demonstrates that the disjoint tokenization has a finer granularity compared to farasa as can be seen from both the vocabulary counts and the total number of tokens in a corpus, that is, it has a lower vocabulary and a higher number of tokens for each corpus as compared to farasa.

We can also notice from Table 2 that the reduction in vocabulary for dotless text as compared to standard Arabic text, in the case of farasa tokenizer, ranges between 20.96% (i.e., Quran corpus) and 32.91% (i.e., Ashaar corpus). On the other hand the reduction for the disjoint tokenizer ranges between 33.6% for Quran and 59.61% for the Aggregated corpus. It is important to note that the reduction for the disjoint tokenizer is even higher as compared to the character tokenization, which is the most fine grained and elementary representation of text.

From both Tables 2 and 1, it can be noticed that rich datasets tend to exhibit smaller V′/V ratios across almost all tokenizations. For example, the Wikipedia dataset

1 `https://github.com/ARBML/tkseem`.

showcases lower V‘/V values for words and disjoint tokenizations compared with the news dataset, and a comparable ratio in farasa tokenization, despite having a considerably larger running text N. This pattern is even more apparent in the poetry dataset, which has the smallest V‘/V ratio. This observation leads to a hypothesis that, asymptotically, V‘ grows slower than V as N grows. In essence, it suggests that, generally, the least frequent vocabulary items tend to share more *rasms* than the more frequent ones, leading to a proportional growth of dotless vocabulary as the volume of running text expands.

To further study this phenomenon and explore the correlation between dotted and dotless vocabulary, we analyzed the top *i*th frequent vocabulary, where *i* ranges from 1% to 100%. In each *i*th iteration, we undot the top *i*th vocabulary and calculate the ratio of the dotless vocabulary count to the dotted count. This analysis was conducted across the aggregated datasets using words, farasa, and disjoint tokenizations and plotted in Figure 6. Examining this plot, it is noticeable that the most significant reduction occurs within disjoint tokenization, where the ratio of dotless to dotted vocabulary size experiences the greatest decrease, starting from approximately a 40% reduction. Furthermore, it is clear that the ratio reduction in word tokenization initially starts at a lower rate but increases as the vocabulary grows to the least frequent ones, eventually surpassing the reduction observed in farasa tokenization. Surprisingly, farasa reduction decreases as

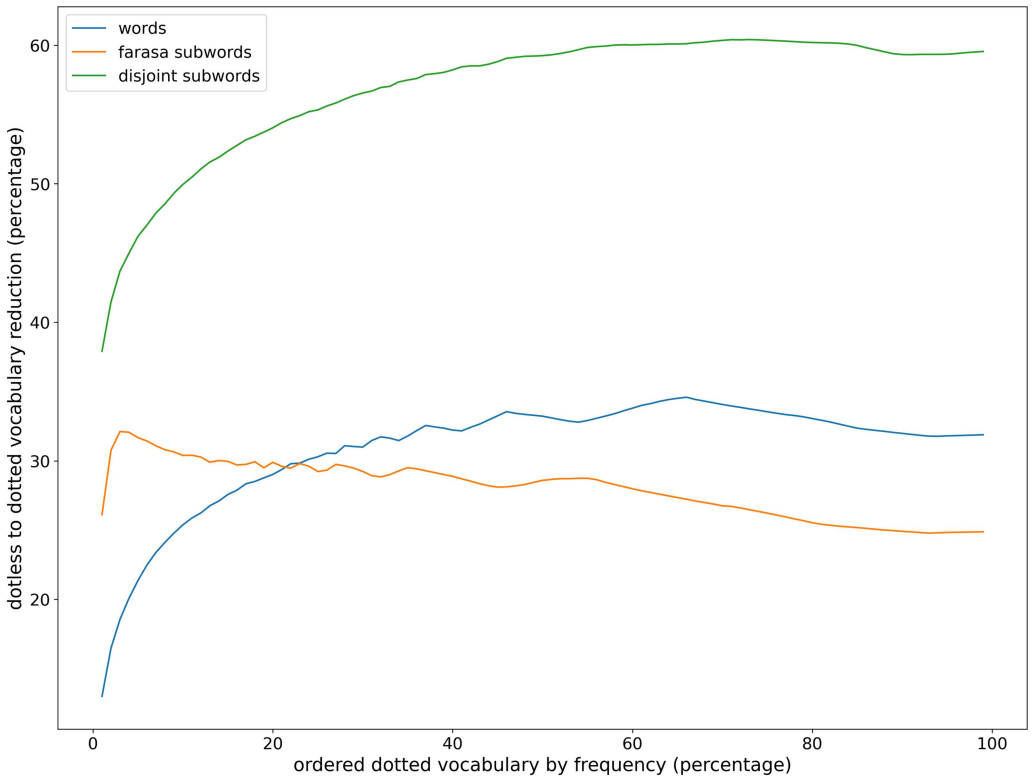


Figure 6
Dotless vocabulary reduction ratio on different tokenization levels.

vocabulary grows in opposition to the other tokenizations. Moreover, across all three tokenizations, the ratio exhibits fluctuations, displaying intermittent rises and falls, forming plateaus stabilizing at the least frequent vocabulary. Based on this analysis, we hypothesize that while many common vocabularies share identical *rasms*, this ratio stabilizes as the vocabulary expands.

The key takeaway from the analysis in this section is that the dotless text leads to significant reduction in vocabulary sizes as compared to the standard Arabic text across all the four tokenizations.

4.2 Zipf's and Heap's Laws

This section introduces the quantitative analysis we performed on dotted and dotless text using Zipf's law (Zipf and Behavior 1949) and Heap's law (Heaps 1978). We are interested to observe the behavior of dotless Arabic text with respect to these well-established relationships, and compare them to standard Arabic texts. Moreover, an in-depth analysis for Arabic corpora is lacking in literature.

Zipf's law can be mathematically formulated as shown in Equation (1), where r indicates the rank of the vocabulary, $F(r)$ represents its frequency, and α is a corpus-specific parameter typically close to 1.

$$F(r) \propto r^{-\alpha} \tag{1}$$

To fit a corpus with Zipf's law and determine the value of α, we applied a log transformation to both sides of the equation, as illustrated in Equation (2). By setting the constant C as the frequency of the most frequent vocabulary within each dataset, we aimed to prevent potential biases when fitting the regression line. This transformation led to the formulation of an equation that exhibits near-linearity on a log-log scale, allowing us to employ linear regression to estimate the slope (α).

$$\log F(r) = -C\alpha \log r \implies \alpha = -\frac{\log F(r)}{C \log r} \tag{2}$$

Figures 7 and 8 plot Zipf's law applied to our datasets using word tokenization. Other tokenizations followed similar patterns to word tokenization. Figure 15 in Appendix A shows the plots for all the tokenizations across all the corpora. Here, we made two plots to reduce the confusion as it is difficult to follow the patterns when plotting all the corpora in one graph.

Analyzing these two graphs, it is clear that word tokenization adheres to Zipf's law for both dotted and dotless text. For other tokenizations, the coarse-grained methods exhibit more adherence to Zipf's law, showing a stronger Zipfian distribution compared with other finer-grained tokenizations. Both graphs here reveal a nearly identical pattern between dotless and dotted plots, with marginal distinctions apparent primarily within the most frequent and least frequent vocabularies.

Heap's law (Heaps 1978) serves as another valuable analytical tool. This law explores the growth of vocabulary within a given corpus relative to its size. This law similarly demonstrates an exponential correlation between the size of the running text

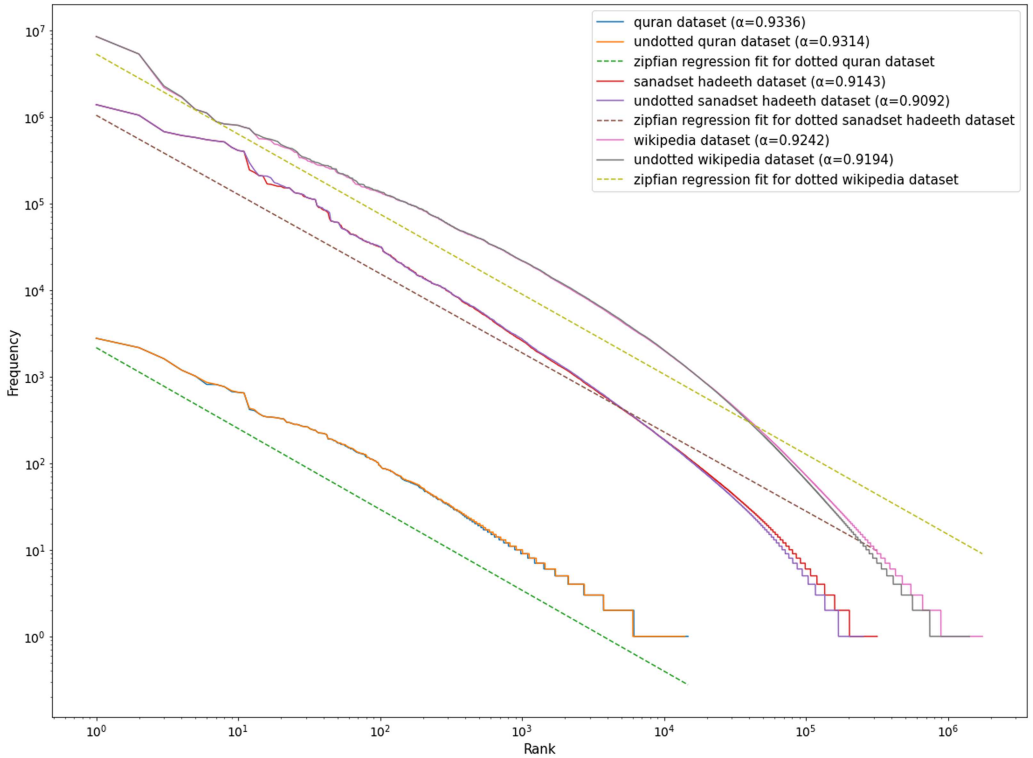


Figure 7
Zipf's law plots for Quran, Sanadset, and Wikipedia datasets comparing dotted and dotless text.

denoted by N and the corresponding vocabulary size denoted by V, governed by an exponent parameter β, typically below 1. This relation is formulated in Equation (3).

$$V \propto N^{\beta} \implies V = kN^{\beta} \tag{3}$$

Following the same procedure conducted for Zipf's law to estimate the parameters using linear regression, we transform both sides of the equation using the log function as in Equation (4). We fit this equation using linear regression to estimate both values k and β.

$$\log V = \log k + \beta \log N \tag{4}$$

Figures 9 and 10 illustrate Heap's law applied to word tokenization across all the corpora. As Quran corpus vocabulary is very small compared with the other corpora, we plotted it separately in a different figure. Other tokenizations follow similar patterns. Figure 16 in Appendix B plots heaps law for all tokenizations across all the corpora.

The depicted plots demonstrate the conformity of both dotted and dotless vocabularies to Heap's law. Notably, the growth of dotless vocabulary consistently trails behind that of the dotted vocabulary across various tokenizations. However, a significant observation arises particularly in the case of disjoint tokenization where the growth rate appears notably slower. This phenomenon suggests a substantial sharing of *rasms*

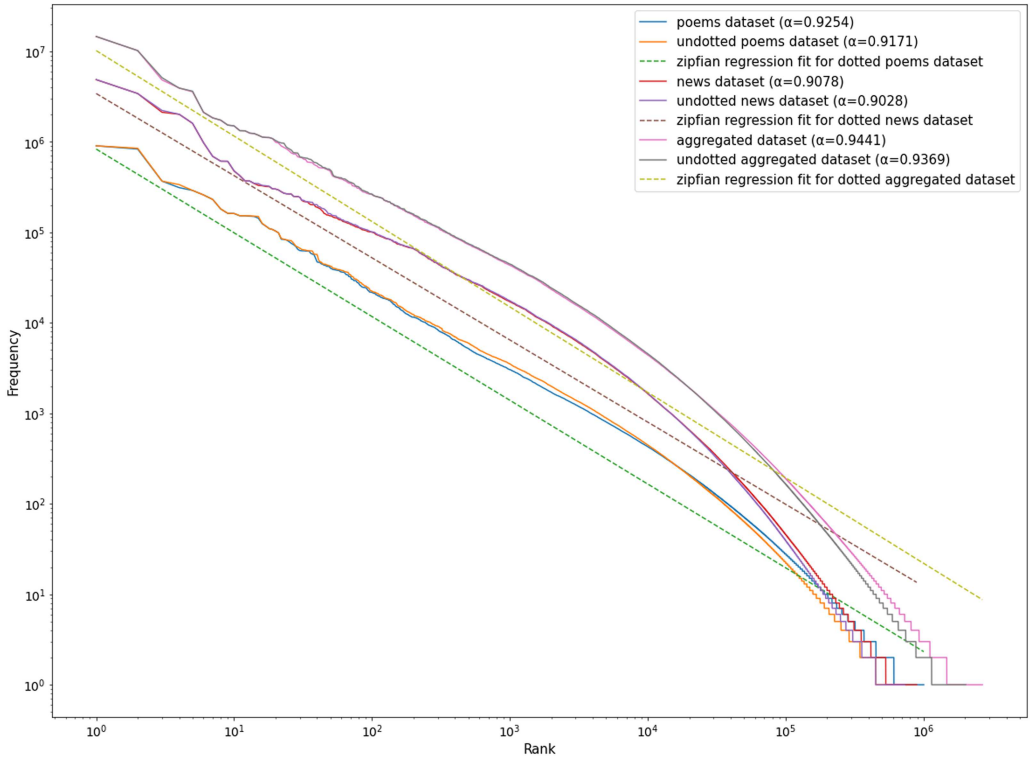


Figure 8
Zipf's law plots for poems, news, and aggregated datasets comparing dotted and dotless text.

among the dotted vocabularies within this specific tokenization scheme, contributing to a slower expansion of the vocabulary.

Another notable observation is the influence of corpus richness on the growth difference between dotted and dotless vocabularies. Specifically, a substantial gap between the growth rates of dotted and dotless vocabularies is noticeable in corpora with higher richness, such as Ashaar and Wikipedia. In contrast, this gap noticeably narrows in less-rich corpora, such as News and Sanadset. This divergence suggests a correlation between corpus diversity and size and the distinctiveness in growth rates between the dotted and dotless vocabularies.

The key takeway from the analysis in this section is that both dotless and standard Arabic text largely follows the Zipf's and Heap's law. However, course-grained tokenization such as word and farasa adhere more strictly to the Zipf's law as compared to a fine-grained tokenizer such as disjoint. Moreover, the vocabulary growth rate is significantly slower for dotless representation, especially for richer corpora having a high ratio of V/N.

4.3 Text Entropy

In information theory, entropy is a measure of uncertainty or unpredictability of a random variable. It quantifies the number of bits necessary to transmit information, that is, the information content of text in our context. In our investigation, we try

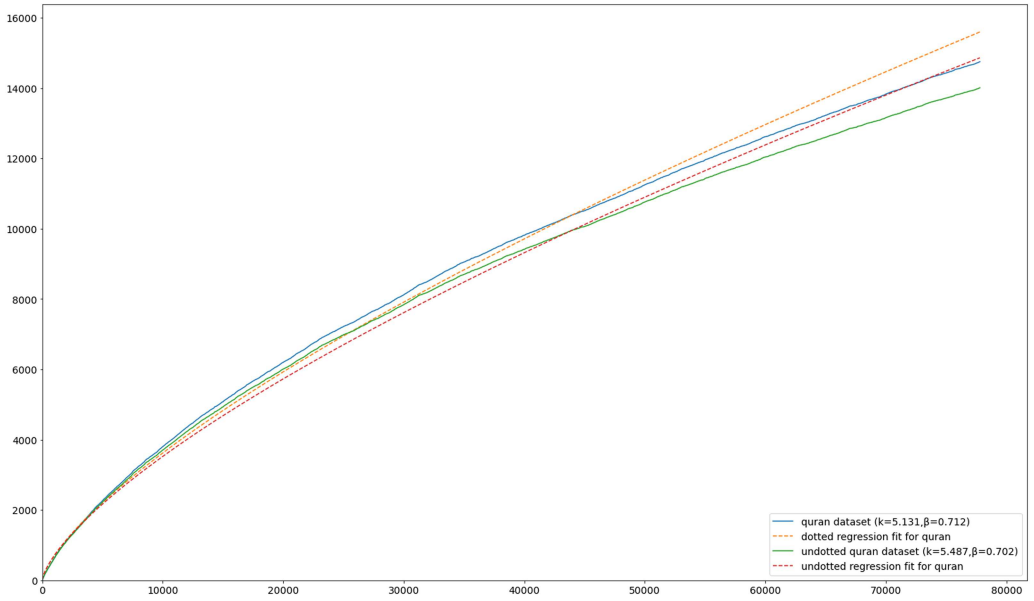


Figure 9
Heap's law plot for the Quran dataset comparing dotted and dotless text.

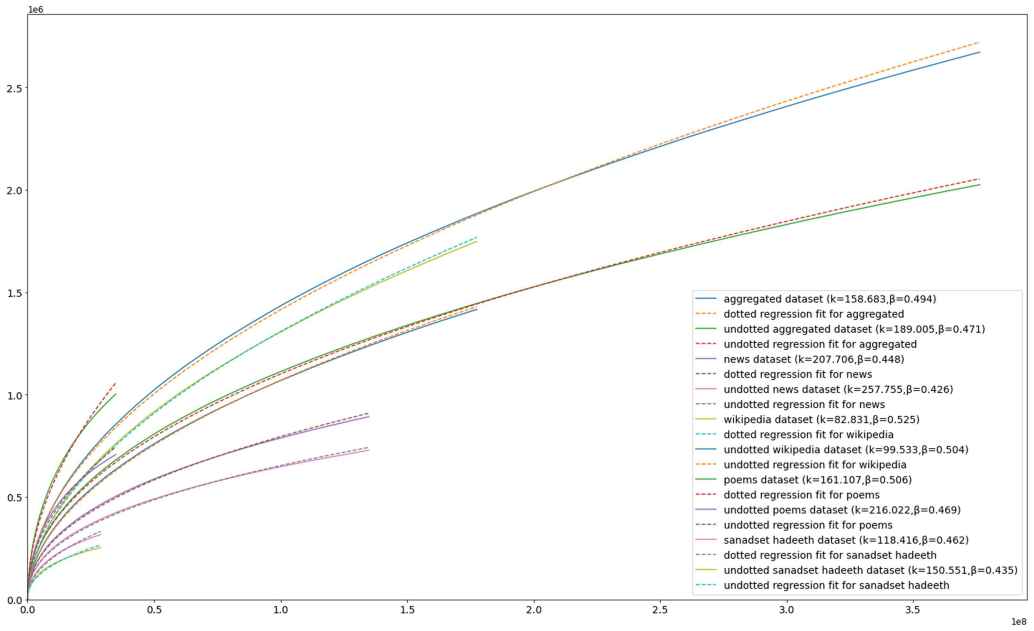


Figure 10
Heap's law plots for Sanadset, Ashaar, Wikipedia, News, and aggregated datasets comparing dotted and dotless text.

Table 3
Entropy results for each tokenization across different text corpora.

Dataset	Words		Farasa		Disjoint		Characters	
	H	H′	H	H′	H	H′	H	H′
Quran	11.02	10.87	8.240	7.968	7.451	6.958	4.15	3.83
Sanadset	10.92	10.67	8.364	8.004	7.644	7.098	4.24	3.86
Ashaar	14.33	13.70	9.659	9.148	8.480	7.687	4.30	3.90
Wikipedia	13.20	12.94	8.892	8.599	8.153	7.493	4.27	3.87
News	13.10	12.87	8.563	8.286	7.981	7.363	4.25	3.85
Aggregated	13.60	13.26	9.063	8.711	8.210	7.523	4.27	3.87

to assess the extent of information loss resulting from the elimination of dots across various tokenization levels for all token strings, denoted as T, in the introduced datasets. Equation (5) was used to compute the entropy of tokens within the corpus where T represents the alphabet, and $P(t)$ represents the probability of token t to appear in the text. Interestingly, this topic has not been extensively explored in the literature for Arabic language. Previous work by Al-Kadi (1998) performed entropy calculations on Arabic text, reporting an entropy of 9.98 bits/words and 2.39 bits/letters. This study considered a corpus made up of only 3,025 words. Table 3 presents the entropy results for each tokenization comparing dotted and dotless text. In this table, H and H' denotes text entropy of dotted and dotless text, respectively.

$$H(T) = \sum_{t}^{T} - P(t) \log_2 P(t) \tag{5}$$

For character tokenization, we can see that the number of dotless characters drops by 12 letters, a reduction of about 39%. However, we can also notice that the highest entropy for dotted characters is 4.3 and the lowest is 4.15, while the highest for the undotted is 3.9 and the lowest is 3.83. If we consider the aggregated dataset as a representation of the language, we can see that the reduction of entropy due to undotting is 0.4.

To compute the upper bound of the entropy at the character level, we assume that characters appear randomly in a string. This can be calculated using Equation (5) as $-\sum_{i}^{31} P(i) * \log_2 P(i) = \log_2 31 = 4.95$ for dotted characters and $\log_2 19 = 4.24$ for undotted ones. Based on this, if we consider the aggregated dataset as a representative set of the language, the language has a characters redundancy of $1 - \frac{4.27}{4.95} = 13.74\%$ for dotted and $1 - \frac{3.87}{4.24} = 8.76\%$ for undotted characters. The redundancy is less for undotted text because dots were helpful in resolving confusion. It can be concluded from this analysis that dots, at most, decrease the text redundancy by 5%. This also indicates that there are other sources of redundancy shared between dotted and dotless text. It is also important to note here that a drop in entropy at the character level between the dotted and dotless version is almost the same across all the corpora. This is understandable, as character frequency distribution is almost the same across different text corpora.

From the table, we can see that Ashaar has the highest entropy at the word level. It is even higher than Wikipedia, which is a magnitude larger in terms of running text

N. This confirms our hypothesis about the vocabulary richness of Ashaar being due to creativity and the use of unusual phrases and sentences. We can also note that Sanadset has the lowest entropy. It is even lower than the Quran dataset, which is a significantly smaller corpus. This indicates that the Sanadset samples share a lot of similarities and the text is quite uniform. It is also important to note here that the drop in entropy at the word level between dotted and dotless text is not uniform across all the text corpora. The highest drop is for the Ashaar corpus and the lowest is for the Quran corpus.

As can be noticed from Table 3, the entropy trend at the subword levels follows similar patterns as the entropy at the word level for both the dotted and dotless version of texts. However, the Quran corpus has the lowest entropy at both the subword levels, whereas it has the second lowest entropy at the word level behind the Sanadset corpus. Moreover, the drop in entropy for the dotless text at the disjoint token level is the highest as compared to other token levels. Thus encoding and representing dotless Arabic text at different token levels can have different implications in text and context understandability and other tasks such as dot retrieval.

4.4 Comparison with English

Arabic has a dense morphology and a rich linguistic structure compared with widely spoken languages like English. This complexity significantly expands Arabic's vocabulary. As shown previously, using dotless text reduces vocabulary by consolidating multiple dotted vocabularies into a single dotless one called *rasm*. To further study and quantify this reduction, we compared dotless and dotted texts in Arabic against English.

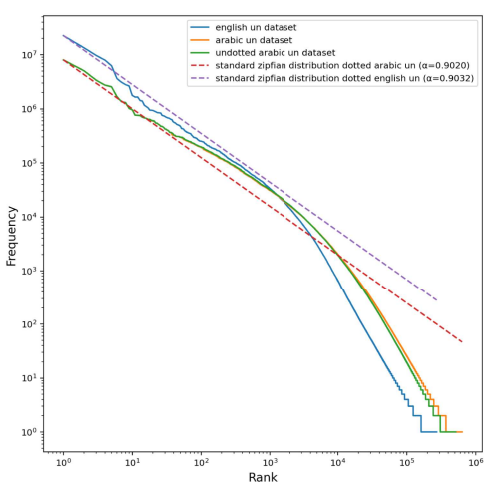
We approached this comparison in two ways. First, we used a parallel corpus, the UN corpus (Ziemski, Junczys-Dowmunt, and Pouliquen 2016). Second, we chose the English Wikipedia corpus and compared it with the Arabic Wikipedia corpus. The English Wikipedia corpus was much larger in terms of running text than the Arabic one. To make the comparison fairer, we curated a subset of English articles whose total text size was similar to the Arabic corpus.

For both comparisons, we processed the English text by converting characters to lowercase and removing characters not in the English set, including numerals and punctuation. Table 4 shows the statistics for these datasets. In this table, V is the vocabulary size, N is the total number of tokens, H_c is the entropy at the character level, and H_w is the entropy at the word level.

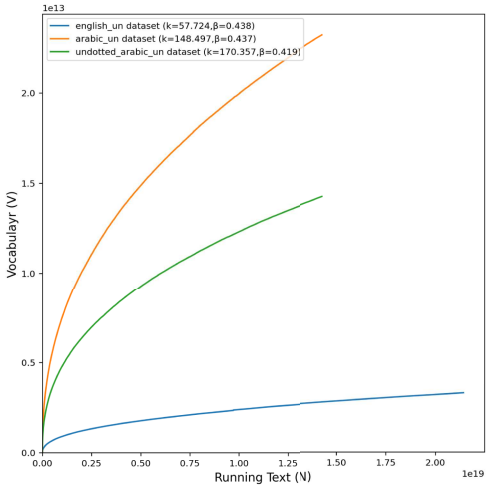
From this table, it can be seen that English has a lower entropy at the character level as compared to dotted Arabic, but dotless Arabic has the lowest. However, dotless

Table 4
Parallel datasets comparison considering dotted and dotless text.

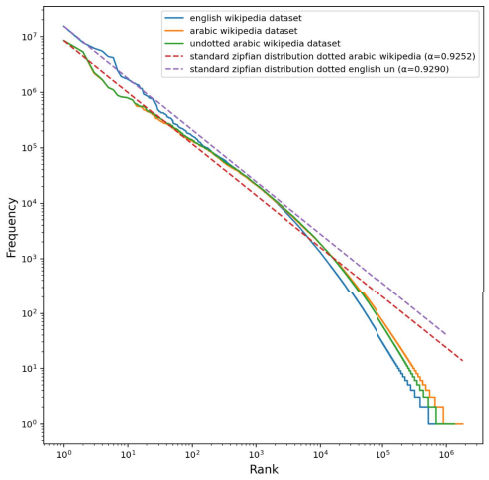
Dataset	N	V	H_c	H_w
English UN	248M	272,567	4.12	9.70
Arabic UN	208M	637,778	4.19	12.21
Dotless Arabic UN		515,946	3.68	12.07
English Wikipedia	198M	1,019,960	4.17	10.97
Arabic Wikipedia	177M	1,811,244	4.27	13.20
Dotless Arabic Wikipedia		1,345,853	3.72	12.93



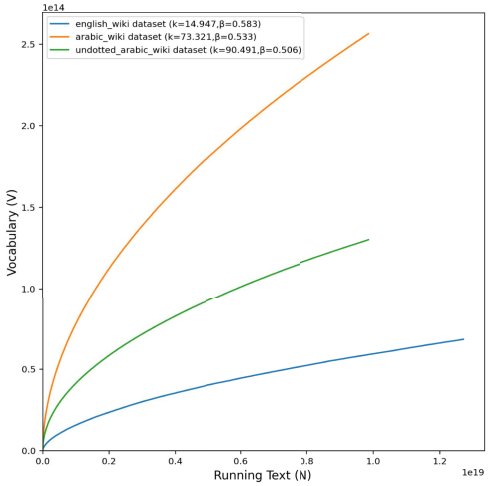
(a) UN parallel Zipf's law



(b) UN parallel Heap's law



(c) Wikipedia Zipf's law



(d) Wikipedia Heap's law

Figure 11
Zipf's and Heap's law for English-Arabic parallel datasets.

Arabic has a higher entropy at the word level as compared to English. It is also interesting to note that although the UN corpus is parallel, the number of words in English is 40M (around 17%) more as compared to Arabic.

To further study vocabulary frequencies and growth, we plot Zipf's and Heap's laws for both corpora in Figure 11. From this graph, we can notice that the Zipfian fit overlaps for both corpora. It is interesting to observe this generalization across languages. It can be noted that in the UN corpus, the plot for English seems to be shifted upwards for the most frequent vocabulary and downwards for the least frequent vocabulary. Following a similar pattern for the Wikipedia corpus, it seems that the most frequent words in English are more frequent than in the case of Arabic. However, the least frequent English words are less frequent than the least frequent Arabic words.

For the vocabulary growth described by Heap's law, it can be seen that Arabic is larger by orders of magnitude in terms of vocabulary, although the English running text is larger for both datasets, UN and Wikipedia. It can also be noted from the graph that the Wikipedia corpus is richer than the UN corpus. Following our previous findings that the vocabulary growth for dotless text is slower for rich datasets, we can see this pattern clearly in the UN corpus as compared to Wikipedia. In fact, in Wikipedia, the dotless text growth is close to the average of both English and Arabic vocabulary growth.

5. Language Modeling

Language modeling stands as a core area of research within NLP, focusing on modeling and generating texts that mimics human language patterns. The primary objective is to predict a token given the context. A common type of language model, termed the causal language model, aims at predicting the next token given a sequence of preceding tokens. These models play a vital role in enhancing the outputs of diverse downstream tasks such as optical character recognition (OCR), speech-to-text (STT), and grammar correction, among others.

Language models are constructed through either statistical methods or deep learning approaches. In statistical methods, text generation relies on determining the most likely token given a preceding sequence of tokens, computed from the corpus to establish conditional probabilities of tokens. These models, often termed n-gram models where n denotes the sequence length of tokens considered, use probabilistic approaches. Conversely, neural language models employ neural networks to predict the most probable token. Notably, statistical language models are typically shallower in structure compared to neural models, offering lower language comprehension and abilites.

Statistical language models face significant challenges in predicting unseen or infrequently used words due to their reliance on word probabilities computed from corpus counts. To mitigate this issue, smoothing techniques have been developed, including Laplace smoothing, discounting, interpolation, backoff, and notably, Kneser-Ney smoothing (Ney, Essen, and Kneser 1994), which is acknowledged as one of the most effective smoothing methods in the literature (Zhang and Chiang 2014).

In this study, we experimented with both of these types of language modeling, that is, the statistical n-grams and neural language models. For neural language models, we experimented with two architectures, RNNs (Chung et al. 2014) and transformers (Vaswani et al. 2017). The architecture setup will be detailed in a separate subsection for each. For dataset splits, each dataset has been split into 90%–10% for training and evaluation, respectively. We further take a validation set of 5% from the training split for model calibration. We selected five different datasets varying in size, domain, and style. It should be noted that all the five datasets consist of classical or modern standard Arabic. The goal was to investigate the behavior of language models generated using dotless Arabic and compare them qualitatively with the language models generated using standard dotted script. We did not use datasets generated with user-generated content such as social media posts and blogs, which can be interesting to investigate. However, such datasets were used for downstream NLP tasks as presented in Section 7.

Language models can be evaluated with two distinct approaches: extrinsic and intrinsic. Extrinsic evaluation involves assessing the model's performance within a downstream task, such as speech recognition or optical character recognition. Conversely, intrinsic evaluation refers to assessing the model's performance without relying on external tasks. Perplexity (PPL) serves as a metric for intrinsic evaluation, quantifying the model's surprise when presented with a given sentence, and leveraging the

Table 5
Train, validation, and test statistics for datasets used in language modeling tasks.

Dataset	Text Type	Train Vocab	Val Vocab	Test Vocab
Quran	Dotted	13,435	1,720	3,161
	Dotless	12,102	1,658	3,018
Sanadset	Dotted	299,162	81,763	121,488
	Dotless	215,758	65,941	95,132
Ashaar	Dotted	881,669	208,236	321,650
	Dotless	560,069	151,916	225,293
Wikipedia	Dotted	970,567	211,025	323,331
	Dotless	730,885	173,672	259,353
News	Dotted	837,244	231,783	333,842
	Dotless	617,593	187,819	263,524

probabilities learned during training. The calculation of perplexity is formulated in Equation (6):

$$\mathrm{PPL}(D, \mathcal{M}) = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log \mathcal{M}(w_i)\right) \tag{6}$$

Where $\mathrm{PPL}(D, \mathcal{M})$ represents the perplexity of the language model $\mathcal{M}$ on a test dataset D, N is the total number of words in the dataset, w_i is the i-th word in the dataset, and $\mathcal{M}(w_i)$ is the probability assigned by the language model $\mathcal{M}$ to the i-th word.

A lower perplexity value indicates that the language model is better at predicting a token given the context. Before discussing the results of the language models, we present the preparation procedure used for the datasets. Table 5 shows the statistics of these datasets for each split in terms of the vocabulary size.

It should be noted that we applied some selection criteria for our datasets as a result of which some samples were eliminated. For instance, in the Ashaar dataset, we were interested in studying classical poetry. Hence, we selected samples where the meter is known to be from the classical poetry meters. Furthermore, the Wikipedia dataset contains samples with varying lengths. To capture meaningful samples, we only selected those with lengths greater than or equal to 30 tokens.

As vocabulary follows a Zipfian distribution, we only select the words that cover 95% of the text, considering the rest as unknowns. It should be noted that subword tokenization was applied as follows. We first selected the words that cover 95% of the running text, then applied the subword tokenization on these words.

5.1 Statistical *n*-grams

In this subsection, we present the results of statistical n-gram language models on dotted and dotless Arabic text. For implementation, we used KenLM toolkit (Heafield 2011). KenLM is an n-gram language model toolkit that implements the modified Kneser-Ney smoothing (Heafield et al. 2013). The toolkit implements disk-based streaming algorithms, allowing it to be seven times faster than its counterpart SriLM (Stolcke

2002), a popular language model toolkit, in estimation with efficient memory management (Heafield 2013). We experimented with different orders of *n*-grams ranging from bigrams to 6 grams. However, other hyperparameters are set to standard values.

Figure 12 presents the *n*-gram counts for each order we investigated in this work. In this figure, for each dataset, the normal line is the dotted *n*-gram counts and the dashed line represents the undotted counts and both lines share the same color. Each row in the figure presents the results of a tokenization with two groups of datasets presented in two separate plots for better readability.

A general observation from this graph is that the difference between the *n*-gram counts between the dotted and dotless text is higher for fine-grained tokens such as characters and disjoint as compared to word and farasa. For *n*-grams at the word level, it has significantly higher counts as compared to other tokens for lower *n*-gram levels but then it plateaus for higher *n*-grams. For subword as tokens, the count is small for lower *n*-grams but increases rapidly for higher *n*-grams, even reaching values comparable to word tokens for 6-grams.

Table 6 presents the OOV statistics for dotted and dotless text after splitting our dataset into training and test sets. The row labeled '*Ratio*' presents the reduction in OOV rates for dotless text as compared to dotted texts. It should be noted that the last column represents the aggregated dataset. OOVs are calculated as follows: As we only considered vocabulary that covers 95% of the running text, other vocabularies are set to unknown. We counted the total occurrences of unknowns and reported the results.

It can be noticed from this table that the reduction in OOV rate between dotless and dotted texts resembles similar patterns for rich datasets as in Tables 1 and 2. For instance, it can be noticed that the poems dataset has a high reduction across all tokenizations as compared to the Sanadset dataset. Wikipedia, although it has more running text as compared to News, it has a similar reduction in OOV. We can also note that for the farasa tokenization the reduction ratios for the dotless texts are mostly lower than other tokenizations. The results in this table seem to follow our conclusions and hypotheses drawn in our discussion in Section 4.

Figure 13 plots the perplexity of our statistical models trained on all the presented datasets across different tokenizations for various *n*-grams. As in the *n*-gram counts figure, the dashed lines represent the dotless version of the dataset. A general pattern to note in this figure is that as the *n*-gram order increases, the difference in perplexity between dotted and dotless text decreases. This pattern becomes more apparent for fine-grained tokenization. We can also note that, for less rich datasets, that is, those that have a smaller V/N, the perplexity for the dotted texts is almost the same as the dotless texts for higher order *n*-grams across various tokenizations even though the vocabulary size is significantly smaller for the dotless versions.

5.2 Neural Language Models

For neural language models, we experimented with both the RNN and transformer architectures. The details for each architecture are described below. However, both architectures share the following training setup.

The learning rate is reduced on plateaus when little progress has been made. That is, we decrease the learning rate by a factor of 0.75 for RNNs and 0.25 for transformers if no improvement was noticed after a full epoch. The training continued for 100 epochs at most, unless no significant improvement with a margin greater than 5×10^{-3} is witnessed for consecutive 5 epochs. The best model that achieved the lowest validation loss is kept. We built the model with PyTorch (Paszke et al. 2019), exploiting the

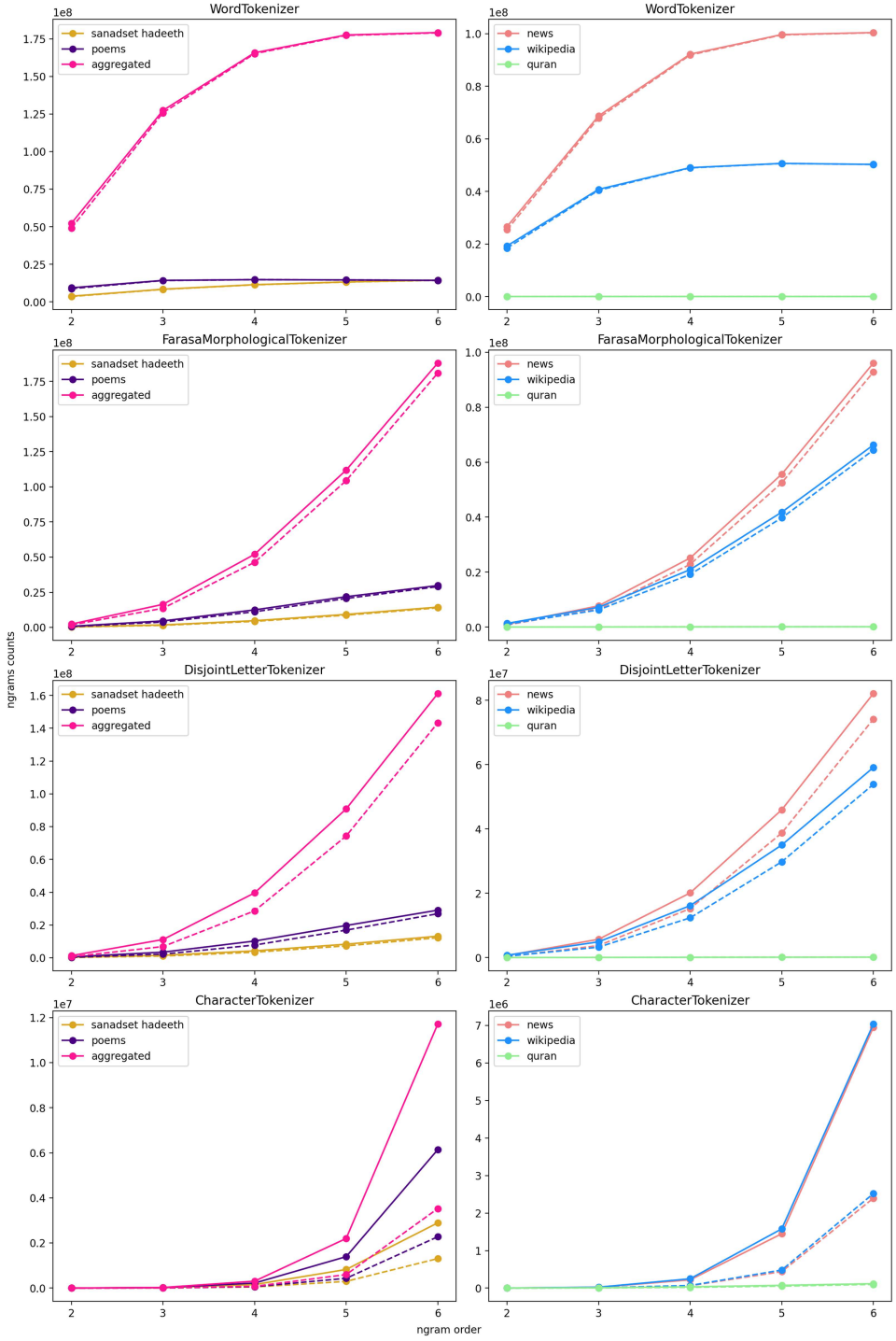


Figure 12
n-gram counts at various tokenization levels across all the datasets.

Table 6
OOV statistics for dotted and dotless texts across different tokenization levels and datasets.

Tokenization		Quran	Sanadset	Poems	Wikipedia	News	Aggregated
Words	***Dotted***	910	13,273	34,577	57,329	39,177	94,216
	Dotless	786	8,619	20,080	40,901	26,507	60,093
	Reduc. (%)	13.63	35.06	41.93	28.66	32.34	36.22
Farasa	***Dotted***	390	4,253	12,514	32,153	17,997	51,268
	Dotless	300	2,948	8,112	25,104	13,680	37,430
	Reduc. (%)	23.08	30.68	35.18	21.92	23.99	26.99
Disjoint	***Dotted***	287	2,534	5,985	7,747	5,619	12,196
	Dotless	185	1,078	2,216	3,116	2,252	4,417
	Reduc. (%)	35.54	57.46	62.97	59.78	59.92	63.78

batteries-included features provided by PyTorch-Lightning package (Falcon et al. 2019). For datasets, we follow a similar setup as in the *n*-gram experiments, partitioning each dataset into 85%, 5%, and 10% splits for training, validation, and test with a batch size of 64 samples.

We fixed the sequence length for each dataset for a given tokenization. We discard the remaining text for samples larger than the specified sequence length and padded samples with smaller sequence lengths. The padding token is ignored in loss and perplexity calculations. We noticed that few samples in different datasets are significantly larger in length as compared to the rest of the samples. Thus, setting the sequence length to the maximum sequence length will lead to lengthy training with little knowledge learned. Additionally, this issue becomes serious with fine-grained tokenization as learning in RNNs is less efficient for longer sequences (Khandelwal et al. 2018; Rumelhart, Hinton, and Williams 1985; Werbos 1990). Hence, we considered different percentiles for different tokenization as the maximum length: 0.99 percentile for word tokenization, 0.975 percentile for disjoint letters tokenization, and 0.95 percentile for character tokenization.

5.2.1 RNN-based Language Model. For the RNN-based language model, the model comprises an embedding layer, a dropout layer (Gal and Ghahramani 2016), a layer of stacked GRU units (Chung et al. 2014), a dense layer utilizing ReLU activation, and another dropout layer followed by a dense output layer with softmax activation to predict the next token as a causal language model. We used cross-entropy loss and Adam (Kingma and Ba 2014) as an optimizer.

As we are experimenting with different datasets from different domains and different tokenizations, the optimal set of hyperparameters may differ from one setup to another. For that reason, we implemented a tuning procedure that determines the best hyperparameters for a given dataset with a given tokenization. It should be noted that this tuning was applied on the dotted experiment; then the resulting best hyperparameters are applied to the dotless system. We used the AHSH scheduling technique (Li et al. 2020) that terminates bad trials early. We used the Ray framework (Liaw et al. 2018), which is a well-known framework for distributed computing. We optimized the hyperparameters in a grid-search style on a subset of the training set.

With word tokenization, or in any experiment involving a large vocabulary, the parameters in the embedding and output layers increase significantly. Research has

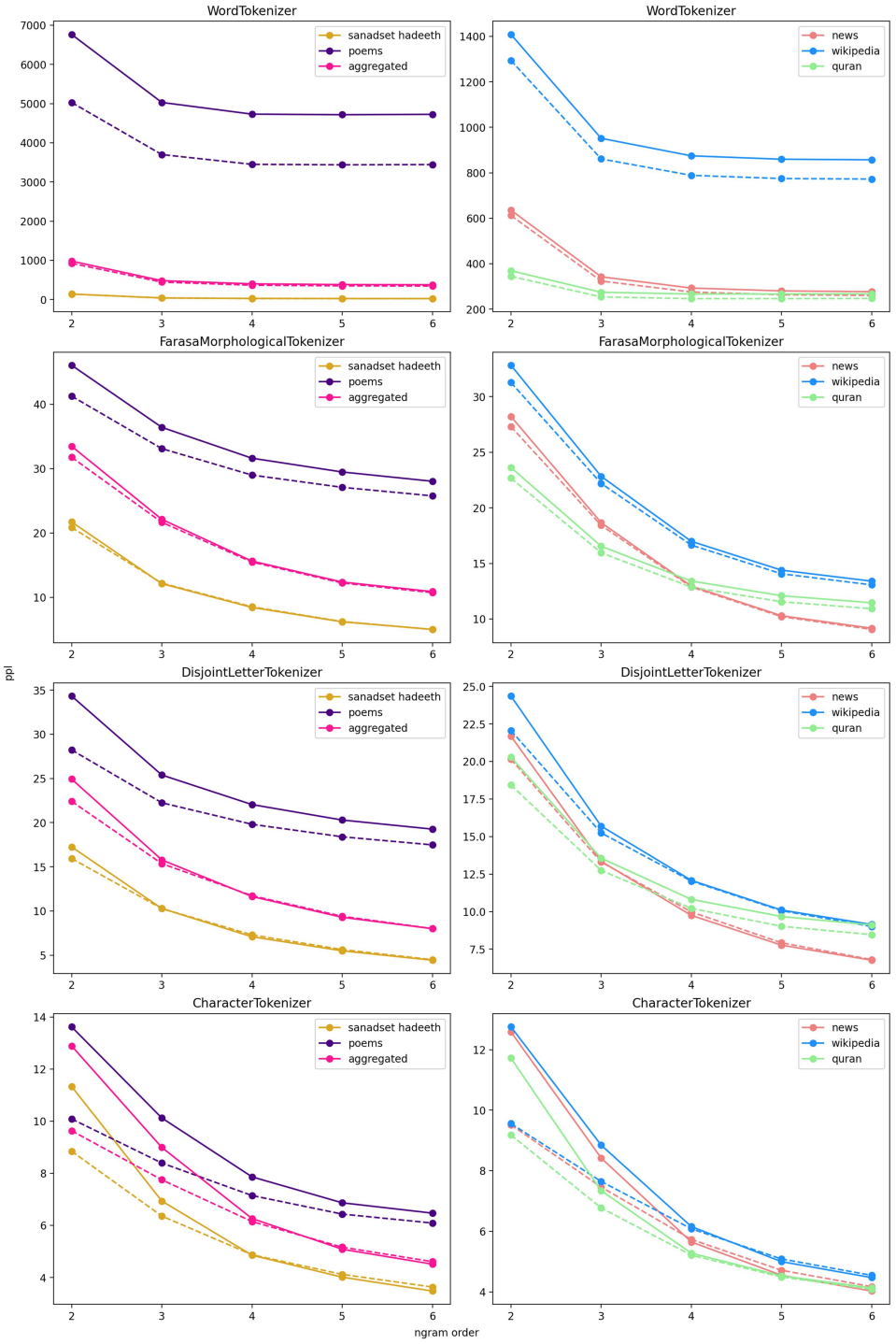


Figure 13
n-gram perplexities at various tokenization levels across all the datasets.

Table 7
Perplexity results using RNN-based language models.

Dataset		Words	Farasa	Disjoint	Characters
Quran	PPL	179.02	22.89	15.76	5.28
	PPL‘	195.69	18.54	14.90	4.66
Sanadset	PPL	33.11	5.16	4.87	2.79
	PPL‘	35.85	5.31	5.11	2.74
Poems	PPL	1,195.47	36.60	18.30	6.17
	PPL‘	1,022.35	31.40	16.80	5.60
News	PPL	166.90	7.72	7.00	3.17
	PPL‘	179.09	8.60	7.29	3.18
Wikipedia	PPL	290.25	9.55	8.79	3.61
	PPL‘	296.16	10.49	9.13	3.52

shown that these layers have similar properties (Press and Wolf 2017; Inan, Khosravi, and Socher 2016; Nowlan and Hinton 1991; Pappas, Miculicich, and Henderson 2018). Consequently, they can share the same parameters, significantly reducing the model size. We used this technique, known as weight tying, when applicable, based on the optimal hyperparameters determined from the tuning procedure.

Table 7 presents the perplexity results of the RNN-based language models. PPL is the perplexity of dotted text while PPL‘ is the perplexity of the dotless text. Analyzing the results from this table, it can be noted that the Poem dataset has significantly high perplexity at the word level, which indicates that predicting the words for poems in Arabic is a very difficult task. Perplexities for other tokenization levels are also comparatively higher for the Poem dataset but still comparable to other datasets. Moreover, the perplexities at the word level are significantly higher across all the datasets as compared to other tokenizations. Another interesting thing to note is that dotless text has higher or comparable perplexities to the dotted counterparts for most of the datasets across different tokenizations. This seems to indicate that the tokens that are merged due to dots removal mostly appear in the text under different contexts. This also implies that correct words can be inferred from the dotless tokens based on the contexts of the sentences.

As the dotless text significantly reduces the vocabulary size as compared to the dotted text, the embedding layer of the dotless model becomes smaller than the dotted model. This results in significant reduction in the model parameters. Figure 14 plots the model parameters of dotted text as compared to the dotless on different datasets across all the tokenizations. The figure shows a noticeable reduction in terms of model size, especially for rich datasets. It can also be noted that the difference in model size is almost negligible for character tokenization. This is because the embedding layer for characters is negligible as compared to other layers due to the small vocabulary size. Another interesting observation arises with farasa tokenization, where, in certain datasets, the model size of the dotted representation is unexpectedly smaller than the dotless one. This occurs because the ratio of dotless to dotted vocabulary decreases as the vocabulary size increases, as shown in Figure 6. These findings highlight the importance of carefully selecting the vocabulary size, particularly with this tokenization, as these choices can subtly impact the performance.

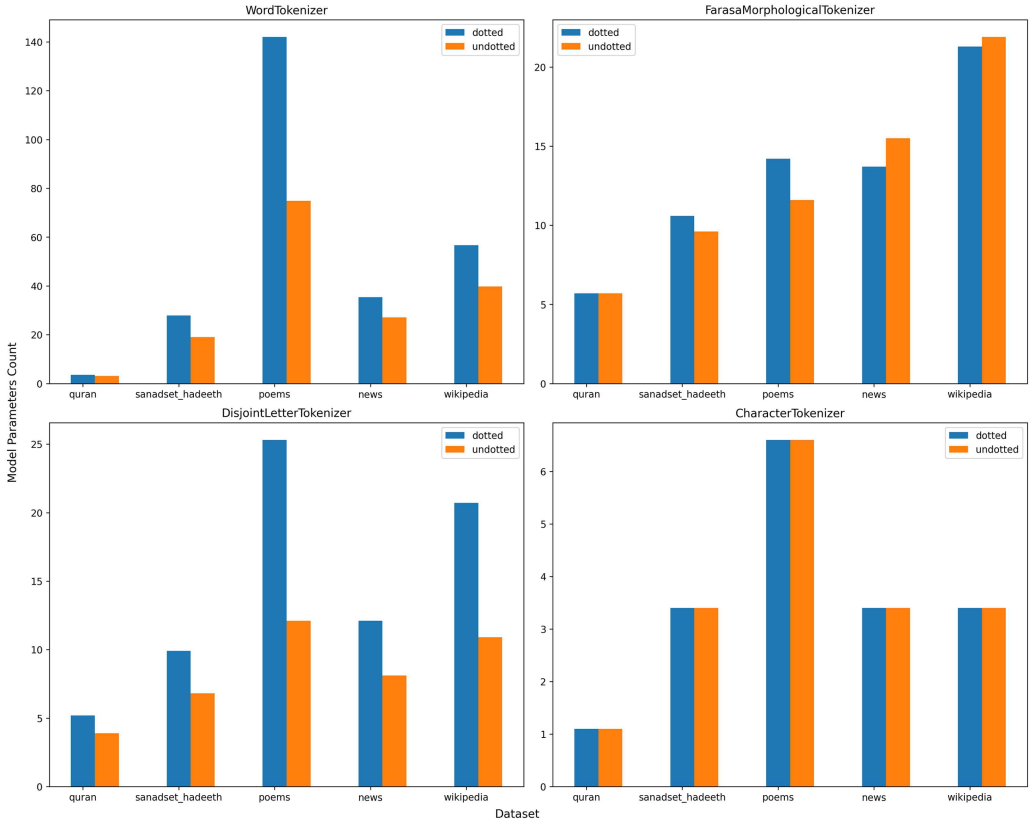


Figure 14
Neural language model parameter size comparison for dotted and dotless text.

5.2.2 Transformer-Based Language Models. Transformers (Vaswani et al. 2017) have emerged as a cutting-edge architecture for generative modeling, originally proposed for machine translation. To adapt this architecture for language modeling, we implemented a model consisting of 3 encoder layers with 8 attention heads and 2,048 feedforward hidden parameters. In line with the original paper, we used cosine-based positional encodings. For optimization, we opted for Rectified Adam (RAdam) (Liu et al. 2020) due to its demonstrated stability during training compared to Adam.

Table 8 presents the results of our experiments with transformer-based language models across all datasets. As with the RNN-based results, PPL is reported for the dotted datasets, while PPL' represents the perplexity for the dotless counterparts.

Analyzing the results presented in this table, a significant decrease in perplexity is evident compared to the RNN-based language models (cf. Table 7). This marked reduction underscores the superior capabilities of transformers over RNNs in modeling natural language and capturing long-term contexts. Notably, analogous patterns observed in the RNN-based experiments are similarly reflected here, affirming the consistency of observed trends across different modeling architectures.

Additionally, the consistent performance improvements achieved by transformers, especially evident when confronted with rich and diverse datasets such as poems or Wikipedia articles, underscore their efficacy in handling complex linguistic context and diverse vocabulary size.

Table 8
Perplexity results using a transformer-based language model.

Dataset		Words	Farasa	Disjoint	Characters
Quran	PPL	116.18	10.35	9.89	4.53
	PPL‘	117.74	11.16	9.80	4.40
Sanadset	PPL	20.72	4.45	4.17	2.55
	PPL‘	22.08	4.71	4.40	2.56
Poems	PPL	908.37	15.14	13.95	4.95
	PPL‘	792.80	16.13	13.26	4.82
News	PPL	106.50	6.53	5.75	2.94
	PPL‘	113.71	7.22	6.20	2.96
Wikipedia	PPL	197.29	8.39	6.53	3.38
	PPL‘	198.11	9.23	7.65	3.35

6. Restoring Dots to Dotless Text

This study introduces dotless Arabic text as an alternative representation to dotted text for Arabic NLP. For tasks requiring Arabic text as output, we need to convert the dotless representation back to the standard Arabic texts having dots for human readability. Accordingly, we extended our work by building a system to retrieve dots for dotless text. The successful implementation of such a system depends on a comprehensive and diverse dataset. Hence, we used the Wikipedia corpus introduced in Section 3 to train this system. The dataset is split such that there are 1,368,742 training samples and 72,040 test samples for evaluation. Furthermore, we set aside 5% of the training set as a validation set for model calibration.

The task is modeled as a character-level sequence labeling task, wherein the input is a sequence of character *rasms* and the output is a sequence of characters as text. For dataset cleaning, symbols appearing fewer than 1,000 times—primarily uncommon unicode characters or symbols from other languages—were excluded. The total number of unique tokens after preprocessing was 128 in the input text, including 19 Arabic character *rasms*, digits, punctuation symbols, and other frequently occurring symbols and 144 characters in the output. For preprocessing, we removed some of the Arabic-specific symbols such as diacritics and the tatweel character. We also standardized numbers into English format (0, 1, 2,. . . , 9). We added a space around punctuation characters and eliminated extra spaces.

The model architecture used for this task consists of an initial embedding layer with a hidden size of 512, followed by two bidirectional LSTM layers. This is followed by a dropout layer with a rate of 0.33 and then a dense layer with softmax activation that predicts the probabilities for the output. Optimization is achieved using cross-entropy loss with the Adam optimizer. The learning rate is set to 0.001. To handle the typical length of Wikipedia samples, the sequence length is set to 500, and training is done with a batch size of 256 samples.

To evaluate the model's performance, we utilized two metrics: Word-Error-Rate (WER) and Character-Error-Rate (CER). WER measures errors at the word level and is computed using Equation (7), where:

- S represents the number of substitutions (words incorrectly recognized or generated).

- D represents the number of deletions (words missing in the output compared to the reference).
- I represents the number of insertions (extra words present in the output compared to the reference).
- N represents the total number of words in the reference.

$$\text{WER} = \frac{S + D + I}{N} \tag{7}$$

On the other hand, CER calculates errors at the character level, following a similar methodology to WER.

The proposed model exhibited a WER of 4.7% and a CER of 1.13% on the test set. Error analysis highlighted that less-frequent words, particularly those associated with named entities or individuals, were more prone to misplacement of predicted dots. Further errors were noted in words featuring multiple variants determined by dot placement, where context played a pivotal role in identifying the correct dot arrangement. This error occurrence may be attributed to the limited comprehension capabilities inherent in character-level tokenization.

7. Downstream NLP Tasks

We extend the exploration of dotless representation of Arabic text by applying it across a diverse array of downstream tasks, each selected for its relevance to both the linguistic peculiarities of the Arabic language and the broader objectives of NLP research. Our investigation spans three main areas: text classification, sequence labeling, and neural machine translation. Within text classification, we experimented with sentiment analysis, news topic classification, and Arabic poetry meter classification. For sequence labeling tasks, we experimented with part-of-speech tagging (POS) and named entity recognition (NER). Lastly, the challenge of machine translation serves as a capstone to this exploration, assessing the model's capacity for maintaining meaning and context between languages considering the dotless representation. It's important to note that these tasks were initially optimized for the best performance with dotted text before evaluating the dotless representation under the same optimal hyperparameters tailored for the dotted scenarios.

7.1 Text Classification

In text classification, we have chosen tasks comparing dotless text representation with dotted, addressing diverse challenges and variations. Sentiment analysis represents the class of binary classification tasks. News topic classification demonstrates adaptability to multi-class scenarios. Arabic poetry meter classification, relying on character tokenization, stands as an instance of Arabic language-specific tasks. This selection aims to compare the versatility of dotless representation against dotted across various tasks and setups. Throughout all the experiments, we evaluated the performance with accuracy. When a dataset is imbalanced, we extend the evaluation to cover micro recall, precision, and F1-score.

Table 9
Sentiment analysis results with three different tokenizations.

Representation	Tokenization	Accuracy	Vocab. Size
dotted	Word	74.70	25,908
dotless		75.56	17,576
dotted	Farasa	74.87	11,759
dotless		73.91	9,970
dotted	Disjoint-Letters	78.04	10,035
dotless		77.34	5,233

7.1.1 Sentiment Analysis. Sentiment analysis, a fundamental NLP task, involves automatically extracting and interpreting emotions, opinions, and attitudes expressed in text, to determine whether the sentiment conveyed is positive or negative. Our study investigates the impact of removing dots in sentiment analysis tasks, utilizing the LABR dataset collected from Goodreads[2] (Aly and Atiya 2013). The dataset includes book reviews with ratings indicating sentiment polarity. A unique characteristic of this dataset is its reflection of real-world usage scenarios from daily Arabic content posted on the Web, encompassing a mixture of MSA and dialects commonly used in day-to-day verbal and social media communication. After preprocessing and filtering for the binary classification task, the dataset contains 9,225 training samples and 2,338 test samples. We experiment with three tokenization methods to evaluate and compare models' performance between dotted and dotless text representations. For vocabulary, we selected vocabulary that covers 90% of the running text following subwords tokenization similar to the one applied in language modeling described in Section 5. We used a RNN-based deep learning architecture for the task. The model architecture includes embedding layers, GRU layers, dropout layers, and dense layers, with binary cross-entropy loss. The Adam optimizer was utilized during training. The model's hyperparameters are tuned to ensure a proper setup for the dotted text. For evaluation, we utilized accuracy as the dataset is balanced.

The test accuracy results shown in Table 9 showcase the impact of different tokenizers on both dotted and dotless texts for sentiment analysis. Notably, finer-grained tokenization generally leads to higher test accuracy across both text types. An important aspect to be noted is the significant reduction in vocabulary size when transitioning from dotted to dotless text, across all the tokenization methods, with a negligible effect on performance. This reduction underscores the efficiency of dotless text in maintaining high levels of accuracy while simplifying the model's complexity. Another effect of having a smaller vocabulary size is that the embedding layer becomes smaller, affecting the overall size of the resulting model. In the case of words as tokens, dotless representation show a slightly better accuracy with almost a third reduction in vocabulary size. For the other two tokenizations, dotless text has slightly lower accuracy. Interestingly, disjoint tokenizer shows almost similar performance with vocabulary reduction for dotless representation by almost 50%.

7.1.2 News Topics Classification. News topic classification entails organizing news articles into thematic categories, aiding in information retrieval and content recommendation.

2 https://www.goodreads.com/.

Table 10
News topics classification results.

Representation	Tokenization	Accuracy	Vocab. Size
dotted	Word	97.81	33,082
dotless		96.83	24,972
dotted	Farasa	97.77	12,280
dotless		97.70	13,010
dotted	Disjoint-Letters	97.77	10,604
dotless		97.70	5,995

This task can be considered as an instance of multi-class text classification. In the Arabic context, this task involves assigning articles to predefined categories like politics and sports based on content, and navigating linguistic nuances specific to Arabic. We utilized the Sanad dataset (Einea, Elnagar, and Al Debsi 2019), comprising nearly 200k articles, and selected the Khaleej newspaper subset for experimentation. Preprocessing procedures mirrored those in language modeling and sentiment analysis tasks. Various tokenization methods were explored, including word, farasa, and disjoint tokenizations, with vocabulary covering 90% of the running text at the word level. Similar to the sentiment analysis task, we used an RNN-based architecture. Model architecture consists of an embedding layer, bidirectional GRU layers, dropout layers, and a dense layer with softmax activation. Hyperparameters were tuned using a range of values for the dotted text, and accuracy is used as the primary evaluation metric as the dataset is balanced.

The test accuracy presented in Table 10 showcases the effects of various tokenization methods on both dotted and dotless texts for the news topic classification task. All the three tokenization shows comparable performances for both the dotted and dotless representations. The vocabulary size reduces by a fourth for the dotless text at the word level and reduces by more than 43% for the dotless text at the disjoint level. For the farasa tokenization, dotless text has slightly more vocabulary size as compared to the dotted text. The results for text classification shows the efficacy of using dotless representation, which can provide similar performances while significantly reducing the vocabulary sizes most of the time.

7.1.3 Poetry Meter Classification. Arabic Poetry meter classification involves categorizing classical Arabic poem verses into predefined classes known as "meters" based on a metric system derived from verse diacritics. Earlier methodologies relied on predefined rules and diacritization for classification, while recent advancements showcased the efficacy of RNNs in autonomously classifying verses without prior diacritization. The Ashaar dataset, introduced in Section 3, was used in this experiment. The dataset was filtered to include only the verses belonging to classical Arabic poem meters. The dataset exhibits an inherent imbalance among classes due to the different popularity of different meters, posing a challenge to deep learning architectures. For tokenization, we used character-level tokenization (Abandah et al. 2015; Al-Shaibani, Alyafeai, and Ahmad 2020) with 10% of the dataset allocated for testing and 5% for validation. For preprocessing, we adapted the processing procedure applied in the language modeling experiments. The trained architecture is a GRU-based architecture with a stack of 5 GRU layers.

Table 11 summarizes the results for classifying meters using both the dotted and dotless text representations using character as tokens. Since the dataset is imbalanced,

Table 11
Meter classification results.

Representation	Accuracy	Recall	Precision	F1-score
Dotted	95.01	86.72	88.42	87.56
Dotless	94.29	84.46	85.89	85.17

we reported recall, precision, and F1-scores. Dotless text shows slightly higher accuracy as compared to dotted text and lower F1-score as compared to dotted text. It should be noted that this small difference in performance was achieved by using only 19 character rasms for dotless text representating a reduction of about 39% when compared to dotted text. Dotted text outperforms dotless by about 2% in Recall and 2.5% in Precision. Consequently, the F1-score also shows a similar trend, with dotted text achieving slightly higher values. These findings indicate strong classification performance for both text representations, with dotted text marginally ahead in performance, highlighting their predictive capabilities in meter classification.

7.2 Sequence Labeling

Sequence labeling is the task of assigning labels to tokens within a sequence. Some notable examples of this class of tasks are POS tagging and NER. We compared the performance of dotless text representation with that of dotted text within the context of these two tasks. This section highlights the details of the experiments conducted, along with the results, discussion, and analysis. For each of these experiments, we report accuracy, precision, recall, and F1-score.

7.2.1 Part-of-Speech Tagging. POS tagging is the task of assigning grammatical categories to words in sentences, labeling them as nouns, verbs, adjectives, and so forth. This process enhances syntactic analysis and aids in sentence comprehension. In our study, we explored the impact of using dotless Arabic text compared with dotted text in the context of this task. Our experimental setup utilized the Arabic subset of the PADT dataset from the Universal Dependencies project (Nivre et al. 2020), which aims to standardize syntactic representations across languages. The dataset features 5,959 training samples, 906 for development, and 674 for testing, and presents challenges due to its class imbalance with 17 different tags. The RNN-based model architecture includes an embedding layer, 5 layers of bidirectional LSTMs, and a dense output layer with softmax activation. The Adam optimizer was used along with cross-entropy loss for model training. We focused on word tokenization, as this tokenization is a standard tokenization for such tasks.

Table 12 presents the results comparing dotted and dotless Arabic text in the POS-tagging task. The dotted representation exhibits a larger vocabulary of 21,908 compared with 18,885 for dotless, representing approximately a 14% reduction in vocabulary size for the dotless representation. Despite this reduction, the differences in accuracy, precision, and recall are marginal. These results suggest that dotless text can effectively address the POS-tagging task with the added benefit of reduced vocabulary size. This reduction could simplify model training and deployment, enhancing computational efficiency without significantly compromising performance.

Table 12
POS-tagging performance for dotted and dotless Arabic text.

Representation	Vocab Size	Accuracy	Precision	Recall	F1-score
Dotted	21,908	94.40	90.50	88.44	89.07
Dotless	18,885	93.39	90.17	86.12	87.50

Table 13
NER performance for dotted and dotless Arabic text.

Representation	Vocab Size	Accuracy	Precision	Recall	F1-score
Dotted	27,282	91.13	72.78	47.03	54.77
Dotless	24,332	91.30	70.81	48.11	55.34

7.2.2 Named-Entity Recognition. NER is tasked with identifying and categorizing specific entities within text, such as names of persons, organizations, and locations. This process enhances comprehension by providing structure to unstructured textual data and supports applications across domains like information retrieval and recommendation engines. In this subsection, we explore the use of dotless Arabic text compared to dotted for NER using the ANERCorp dataset, which includes 3,972 training samples and 924 testing samples, with an additional 10% reserved for development. Our experimental setup used a word-based tokenization approach similar to the POS-tagging task. The RNN-based architecture includes an embedding layer, three layers of bidirectional LSTMs with a 0.5 dropout, and a dense layer with softmax for generating class probabilities. Optimization was performed using the Adam optimizer and cross-entropy loss.

Table 13 compares dotted and dotless Arabic text in the NER task. Unlike in POS tagging, dotless text shows marginal improvements in recall and F1-score. The dotted text has a larger vocabulary of 27,282 words compared to 24,332 for dotless. These results suggest that dotless text is a viable alternative for named-entity recognition tasks. It can provide similar performances as compared to dotted text while providing reduction in vocabulary size.

7.3 Neural Machine Translation

Machine translation (MT) is the process of converting text or speech from one language to another. This task is crucial for breaking down language barriers and facilitating global communication across diverse groups. MT systems rely heavily on understanding the nuances of language, including grammar, syntax, and context, to produce accurate and coherent translations. Analyzing the performance of dotted text against dotless is of great importance to demonstrate the capabilities of dotless text in representing the language structure and intricacies. In this subsection, we experimented with English-Arabic translation in both directions.

We utilized a transformer-based architecture (Vaswani et al. 2017) comprising 6 encoder and decoder layers with 512 embedding size and a 2,048-dimensional feed-forward layer, utilizing the RAdam optimizer (Liu et al. 2020). We trained our model on the English-Arabic subset of the iwslt2017 dataset (Cettolo et al. 2017), which includes 231,713 training samples, and 888 validation samples. For testing, we used the 2015 subset containing 8,583 samples. Data preprocessing involved removing uncommon

Unicode characters, normalizing punctuation using sacreMoses (hpl), and unifying numerical characters to English formats. For Arabic, diacritics and tatweel characters were removed, and spaces were added around punctuation. We evaluated the performance using sacreBLEU (Post 2018) on both dotless and dotted texts, with dotless experiments involving the re-addition of dots to predicted texts. To achieve better performance, we averaged the last 10 checkpoints and reported its score if it outperformed the best checkpoint score. For the dot restoration, we utilized a similar system to the one introduced in Section 6 and trained it on this translation dataset to ensure domain adaptation. Batch sizes were adjusted to handle 4k tokens per iteration.

In both translation directions, we experimented with word-based and data-driven subword tokenization using SentencePiece (Kudo and Richardson 2018), which is well-known for such tasks. For word-based tokenization, we included all the vocabulary. This results in vocabulary size was 97,387 for dotted text and 81,272 for dotless text, achieving a reduction of around 16.5%. For subword tokenization, we set the vocabulary size to 4,000. In the case of Arabic text, we observed that this impacts the sequence length, resulting in 81 tokens for dotless text and 89 tokens for dotted text, as we selected sequence lengths covering the full text of at least 99% of the training samples.

7.3.1 English to Arabic. Table 14 presents the results of machine translation comparing dotted and dotless text representations. For each tokenization method, the undotted representation compares the undotted predictions with the undotted version of the test set. Additionally, we restored dots to the undotted predictions using the dot restoration system introduced and reported the corresponding BLEU scores. The results indicate that with coarse-grained tokenization, such as word-based tokenization, dotless text produces results that are on par with dotted text, even after restoring dots. For fine-grained tokenization like SentencePiece, a small difference is observed when dots are restored. Nonetheless, the dotless text maintains robust performance on the undotted text, demonstrating its efficacy in dotless settings. It should be noted that the reduction in the BLUE score for dotless representation is the result of the dots restoration system and not due to using dotless text in the translation task.

7.3.2 Arabic to English. The results in Table 15 illustrate the BLEU scores for Arabic- to English machine translation across word and subword tokenization methods. As in the English-to-Arabic translation, in the undotted version, we removed the dots from the targets before computing the BLEU score. Interestingly, the undotted representation outperforms its dotted counterpart when using word-based tokenization. However,

Table 14
BLEU scores of English-to-Arabic translation comparing dotted and dotless text performance with words and SentencePiece tokenizations.

Representation	Tokenization	BLEU Score
Dotted		12.57
Undotted	Word	13.55
After Dots Restoration		12.48
Dotted		17.38
Undotted	SentencePiece	17.35
After Dots Restoration		16.01

Table 15
BLEU scores of Arabic-to-English translation comparing dotted and dotless text performance with words and SentencePiece tokenizations.

Representation	Tokenization	BLEU Score
Dotted	Word	25.85
dotless		27.10
Dotted	SentencePiece	31.48
dotless		29.77

in the case of the more granular SentencePiece tokenization, the performance of the undotted text decreases to 29.77 as compared to 31.48 for the dotted text. This variation suggests that the finer segmentation of SentencePiece may be more sensitive to the nuances introduced by dots.

As a summary for the downstream NLP tasks, we performed 7 different tasks using various tokenization schemes comparing the standard dotted text with dotless Arabic text representations. The performances using both the representations were comparable across different tokenizations, with dotted representation achieving slightly better performances many times and dotted representation outperforming at other occasions. Dotless representation achieves these results with significant reduction in vocabulary sizes, and in some cases showing reduction of up to 50%. This concludes our findings that dotless representation is a promising and viable approach for Arabic NLP.

8. Conclusion and Future Directions

In this study, we explored the feasibility and adaptability of a novel text representation of Arabic for various NLP tasks that relies on using dotless Arabic *rasms* to represent the Arabic text, thereby significantly reducing the vocabulary size. To examine the capabilities of this representation, we conducted an in-depth analysis comparing dotted and dotless representations across different datasets and tokenizations. We also evaluated the performance of both dotted and dotless texts across various tasks, including language modeling and a wide range of downstream tasks, such as text classification, sequence labeling, and machine translation. Our comprehensive experiments and evaluations have led to several key findings that highlight the potential benefits and implications of adopting this novel representation. For tasks with a generative nature, we proposed a dotting system that restores dots for dotless text.

We noted that the dotless representation significantly reduces the vocabulary size across all tokenization levels, which directly translates into more efficient processing and compact models. Our analysis using Zipf's and Heap's laws confirms that dotless text follows similar patterns compared to its dotted counterpart, demonstrating its viability as a reliable alternative representation. Furthermore, entropy analysis indicates that dotless text maintains a level of linguistic information close to dotted text, with a minimal reduction in information content.

In language modeling tasks, statistical n-gram models and neural language models using RNNs and transformers revealed that dotted and dotless text share similar perplexity patterns. This suggests that, despite the absence of dots, the dotless representation can effectively capture the linguistic structure of Arabic. Notably, the neural models also benefited from the reduction in vocabulary size, leading to smaller model size.

In downstream NLP tasks, our results across sentiment analysis, news topic classification, and Arabic poetry meter classification show that the performance gap between dotted and dotless text is minimal. For sequence labeling tasks such as part-of-speech tagging and named entity recognition, the dotless representation continues to perform comparably well. Finally, in machine translation tasks, dotless text proved capable of achieving on-par results compared to its dotted counterpart in English-Arabic and Arabic-English translation tasks.

Future work could explore several directions to further enhance the practical applications of dotless representation. This includes investigating the integration of dotless text to improve the performance of language models, especially in large language modeling domains. Exploring hybrid methods that combine the dotless and dotted representations for specialized NLP tasks, where context-sensitive dotting might be necessary, would also be valuable. Moreover, investigating the impact of dotless text on speech recognition systems is an interesting research direction to explore. These directions will pave the way for fully leveraging the efficiency and performance benefits offered by dotless representation in real-world applications.

Appendix A

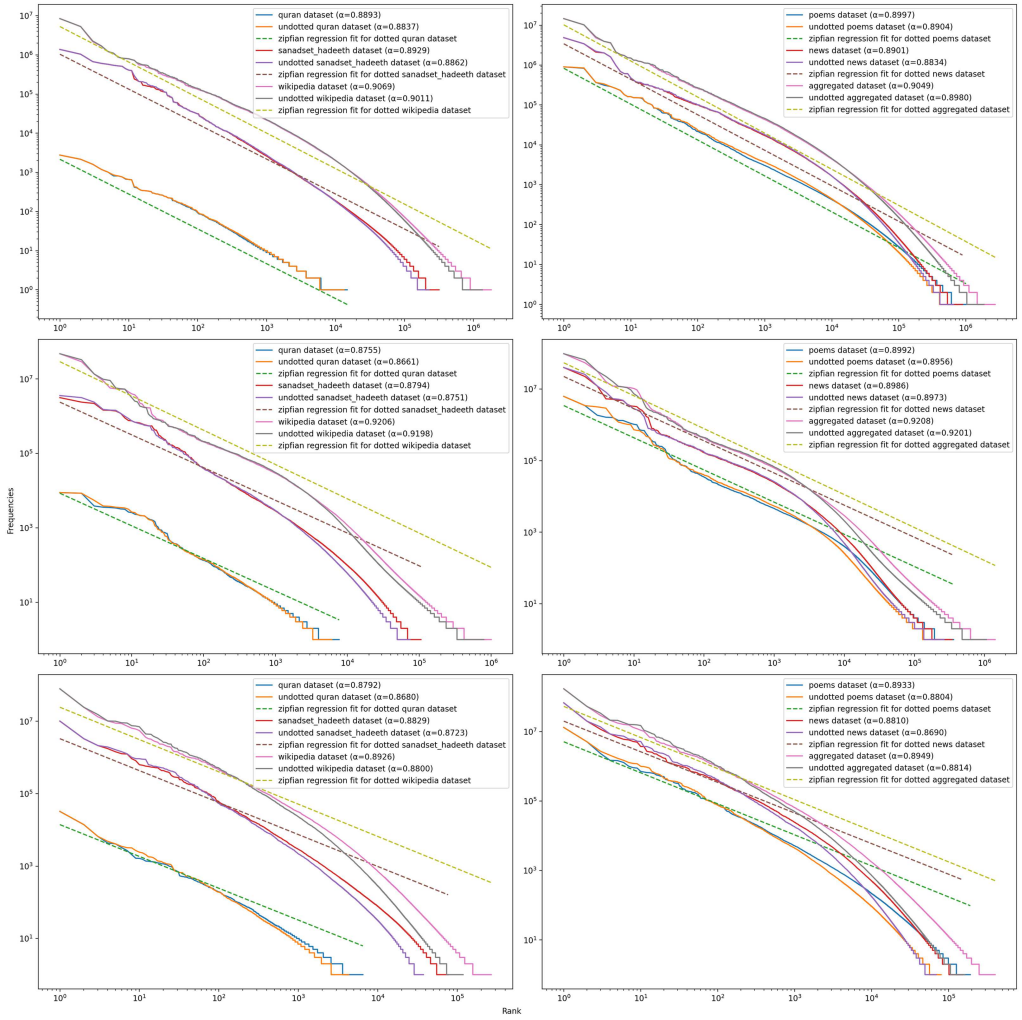


Figure 15
Zipf's Law plots for various datasets with different tokenizations. Each row represents a tokenization level. The first row is for word tokenization, the second is for farasa tokenization, and the third is for disjoint-letters tokenization. Each tokenization is split into two graphs as it is difficult to plot all datasets in one graph.

Appendix B

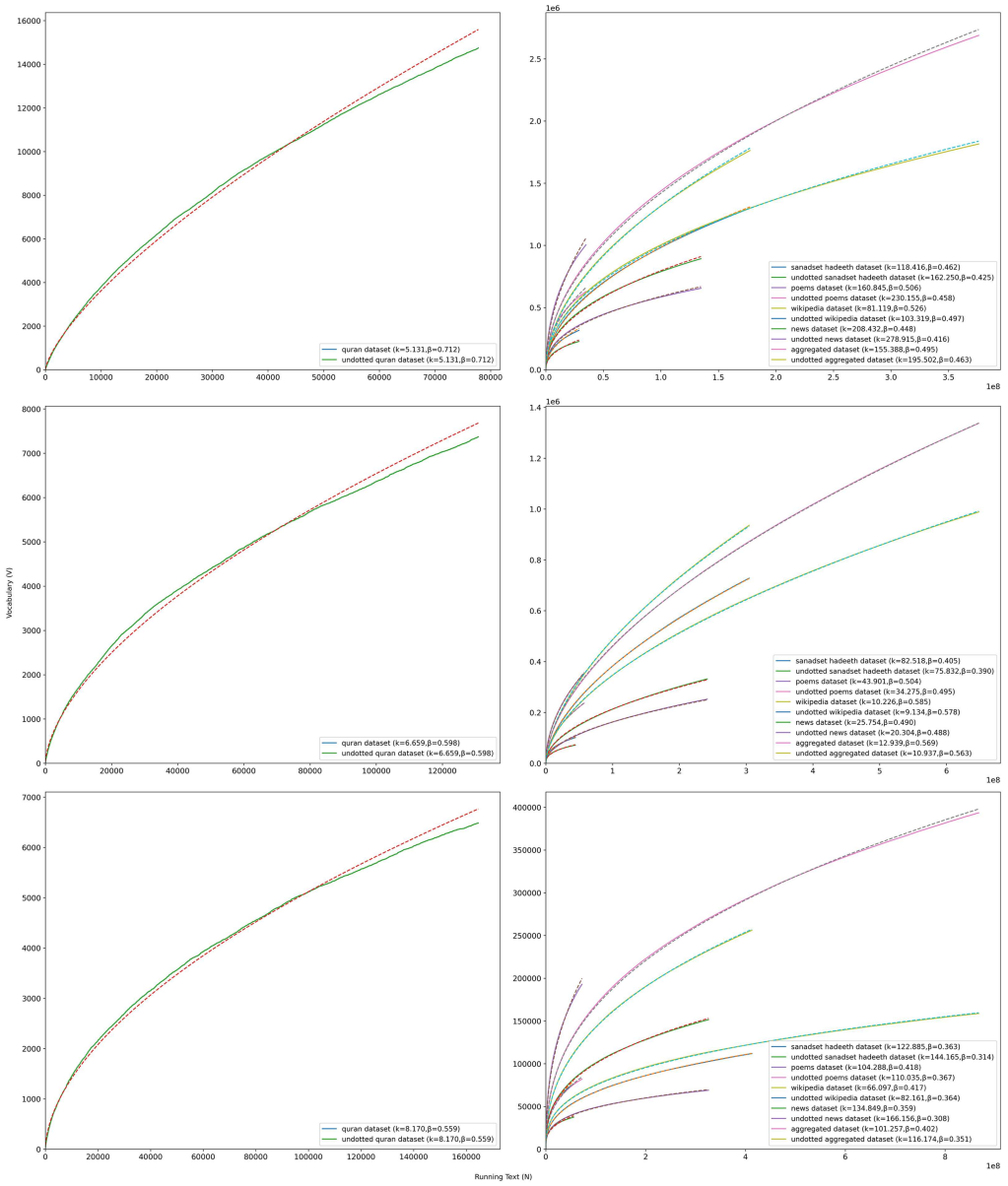


Figure 16
Heap's Law plots for various datasets with different tokenizations. Each row represents a tokenization level. The first row is for word tokenization, the second is for farasa tokenization, and the third is for disjoint-letters tokenization. In each tokenization, Quran has been split into a separate graph as it has a small vocabulary size compared with the other datasets.

Acknowledgments

The authors would like to thank Saudi Data and AI Authority (SDAIA) and King Fahd University of Petroleum and Minerals (KFUPM) for supporting this work through SDAIA-KFUPM Joint Research Center for Artificial Intelligence grant number JRC–AI–RFP–10.

References

hplt-project/sacremoses: Python port of Moses tokenizer, truecaser and normalizer. https://github.com/hplt-project/sacremoses. (Accessed on 04/12/2024).

Abandah, Gheith A., Alex Graves, Balkees Al-Shagoor, Alaa Arabiyat, Fuad Jamour, and Majid Al-Taee. 2015. Automatic diacritization of Arabic text using recurrent neural networks. *International Journal on Document Analysis and Recognition (IJDAR)*, 18:183–197. https://doi.org/10.1007/s10032-015-0242-2

Abdelali, Ahmed, Kareem Darwish, Nadir Durrani, and Hamdy Mubarak. 2016. Farasa: A fast and furious segmenter for Arabic. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 11–16. https://doi.org/10.18653/v1/N16-3003

Abushariah, Mohammad A. M., Raja N. Ainon, Roziati Zainuddin, Moustafa Elshafei, and Othman O. Khalifa. 2010. Natural speaker-independent Arabic speech recognition system based on hidden Markov models using sphinx tools. In *International Conference on Computer and Communication Engineering (ICCCE'10)*, pages 1–6. https://doi.org/10.1109/ICCCE.2010.5556829

Al-Kadi, Ibrahim A. 1998. Study of information-theoretic properties of Arabic based on word entropy and Zipf's law. *Journal of King Saud University-Computer and Information Sciences*, 10:1–14. https://doi.org/10.1016/S1319-1578(98)80001-3

Al-Shaibani, Maged and Irfan Ahmad. 2023. Consonant is all you need: A compact representation of English text for efficient NLP. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 11578–11588. https://doi.org/10.18653/v1/2023.findings-emnlp.775

Al-Shaibani, Maged S., Zaid Alyafeai, and Irfan Ahmad. 2020. Meter classification of Arabic poems using deep bidirectional recurrent neural networks. *Pattern Recognition Letters*, 136:1–7. https://doi.org/10.1016/j.patrec.2020.05.028

Alajrami, Ahmed, Katerina Margatina, and Nikolaos Aletras. 2023. Understanding the role of input token characters in language models: How does information loss affect performance? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9085–9108. https://doi.org/10.18653/v1/2023.emnlp-main.563

Alarifi, Abdulrahman, Mansour Alghamdi, Mohammad Zarour, Batoul Aloqail, Heelah Alraqibah, Kholood Alsadhan, and Lamia Alkwai. 2012. Estimating the size of Arabic indexed web content. *Scientific Research and Essays*, 7(28):2472–2483. https://doi.org/10.5897/SRE11.1708

Alhathloul, Zainab and Irfan Ahmad. 2022. Automatic dottization of Arabic text (Rasms) using deep recurrent neural networks. *Pattern Recognition Letters*, 162:47–55. https://doi.org/10.1016/j.patrec.2022.09.001

Almeman, Khalid and Mark Lee. 2013. Automatic building of Arabic multi dialect text corpora by bootstrapping dialect words. In *2013 1st International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*, pages 1–6. https://doi.org/10.1109/ICCSPA.2013.6487247

Alotaiby, Fahad, Ibrahim Alkharashi, and Salah Foda. 2009. Processing large Arabic text corpora: Preliminary analysis and results. In *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, pages 78–82.

Alotaiby, Fahad, Salah Foda, and Ibrahim Alkharashi. 2014. Arabic vs. English: Comparative statistical study. *Arabian Journal for Science and Engineering*, 39:809–820. https://doi.org/10.1007/s13369-013-0665-3

Aloufi, Khalid. 2019. Quran dataset.

Aly, Mohamed and Amir Atiya. 2013. LABR: A large scale Arabic book reviews dataset. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 494–498.

Alyafeai, Zaid and Maged Al-Shaibani. 2020. ARBML: Democritizing Arabic natural language processing tools. In *Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS)*, pages 8–13. https://doi.org/10.18653/v1/2020.nlposs-1.2

Alyafeai, Zaid, Maged S. Al-shaibani, Mustafa Ghaleb, and Irfan Ahmad. 2023. Evaluating various tokenizers for Arabic

text classification. *Neural Processing Letters*, 55(3):2911–2933. https://doi.org/10.1007/s11063-022-10990-8

Baevski, Alexei, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460.

Bengio, Yoshua, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. In *Advances in Neural Information Processing Systems*, 13:932–938.

Bentouati, Abdelkader. 2020. Arabic language controls before the descent of the holy quran, points - shape - sizes. *Bidayat*, 2(1):18–28.

Boudad, Naaima, Rdouan Faizi, Rachid Oulad Haj Thami, and Raddouane Chiheb. 2018. Sentiment analysis in Arabic: A review of the literature. *Ain Shams Engineering Journal*, 9(4):2479–2490. https://doi.org/10.1016/j.asej.2017.04.007

Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Cettolo, Mauro, Marcello Federico, Luisa Bentivogli, Jan Niehues, Sebastian Stüker, Katsuhito Sudoh, Koichiro Yoshino, and Christian Federmann. 2017. Overview of the IWSLT 2017 evaluation campaign. In *Proceedings of the 14th International Conference on Spoken Language Translation*, pages 2–14.

Chernyavskiy, Anton, Dmitry Ilvovsky, and Preslav Nakov. 2021. Transformers: "The end of history" for natural language processing? In *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Proceedings, Part III 21*, pages 677–693. https://doi.org/10.1007/978-3-030-86523-8_41

Chung, Junyoung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Dehghani, Mostafa, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2018. Universal transformers. *arXiv preprint arXiv:1807.03819*.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Diab, Mona, Mahmoud Ghoneim, and Nizar Habash. 2007. Arabic diacritization in the context of statistical machine translation. In *Proceedings of Machine Translation Summit XI: Papers*.

Einea, Omar, Ashraf Elnagar, and Ridhwan Al Debsi. 2019. Sanad: Single-label Arabic news articles dataset for automatic text categorization. *Data in Brief*, 25:104076. https://doi.org/10.1016/j.dib.2019.104076, PubMed: 31440535

El-Khair, Ibrahim Abu. 2016. 1.5 billion words Arabic corpus. *ArXiv*, abs/1611.04033.

Falcon, William et al. 2019. PyTorch lightning. *GitHub. Note*. https://github.com/PyTorchLightning/pytorch-lightning. 3(6):11.

Gal, Yarin and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. *Advances in Neural Information Processing Systems*, 29:1019–1027.

Guellil, Imane, Houda Saâdane, Faical Azouaou, Billel Gueni, and Damien Nouvel. 2021. Arabic natural language processing: An overview. *Journal of King Saud University-Computer and Information Sciences*, 33(5):497–507. https://doi.org/10.1016/j.jksuci.2019.02.006

Heafield, Kenneth. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197.

Heafield, Kenneth. 2013. *Efficient Language Modeling Algorithms with Applications to Statistical Machine Translation*. Ph. D. thesis, Carnegie Mellon University.

Heafield, Kenneth, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 690–696.

Heaps, Harold Stanley. 1978. *Information Retrieval, Computational and Theoretical Aspects*. Academic Press.

Hochreiter, Sepp and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735, PubMed: 9377276

Inan, Hakan, Khashayar Khosravi, and Richard Socher. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv preprint arXiv:1611.01462*.

Katz, Slava. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3):400–401. https://doi.org/10.1109/TASSP.1987.1165125

Khandelwal, Urvashi, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 284–294. https://doi.org/10.18653/v1/P18-1027

Khreisat, Laila. 2009. A machine learning approach for Arabic text classification using *n*-gram frequency statistics. *Journal of Informetrics*, 3(1):72–77. https://doi.org/10.1016/j.joi.2008.11.005

Kingma, Diederik P. and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kneser, Reinhard and Hermann Ney. 1995. Improved backing-off for M-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, 1:181–184. https://doi.org/10.1109/ICASSP.1995.479394

Kudo, Taku and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71. https://doi.org/10.18653/v1/D18-2012

Li, Liam, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. 2020. A system for massively parallel hyperparameter tuning. In *Proceedings of Machine Learning and Systems*, 2:230–246.

Li, Wentian. 1992. Random texts exhibit Zipf's-law-like word frequency distribution. *IEEE Transactions on Information Theory*, 38(6):1842–1845. https://doi.org/10.1109/18.165464

Liaw, Richard, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E. Gonzalez, and Ion Stoica. 2018. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*.

Liu, Liyuan, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2020. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*.

Magueresse, Alexandre, Vincent Carles, and Evan Heetderks. 2020. Low-resource languages: A review of past work and future challenges. *arXiv preprint arXiv:2006.07264*.

Mghari, Mohammed, Omar Bouras, and Abdelaaziz El Hibaoui. 2022. Sanadset 650K: Data on Hadith narrators. *Data in Brief*, 44. https://doi.org/10.1016/j.dib.2022.108540, PubMed: 36065202

Moreno-Sánchez, Isabel, Francesc Font-Clos, and Álvaro Corral. 2016. Large-scale analysis of Zipf's law in English texts. *PloS One*, 11(1):e0147073. https://doi.org/10.1371/journal.pone.0147073, PubMed: 26800025

Ney, Hermann, Ute Essen, and Reinhard Kneser. 1994. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech & Language*, 8(1):1–38. https://doi.org/10.1006/csla.1994.1001

Nivre, Joakim, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4034–4043.

Nowlan, Steven and Geoffrey E Hinton. 1991. Adaptive soft weight tying using gaussian mixtures. *Advances in Neural Information Processing Systems*, 4:993–1000.

Pappas, Nikolaos, Lesly Miculicich, and James Henderson. 2018. Beyond weight tying: Learning joint input-output embeddings for neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 73–83. https://doi.org/10.18653/v1/W18-6308

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin,

Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32:8026–8037.

Post, Matt. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191. https://doi.org/10.18653/v1/W18-6319

Press, Ofir and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163. https://doi.org/10.18653/v1/E17-2025

Rom, Aviad and Kfir Bar. 2021. Supporting undotted Arabic with pre-trained language models. In *Proceedings of the 4th International Conference on Natural Language and Speech Processing (ICNLSP 2021)*, pages 89–94.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. 1985. Learning internal representations by error propagation. University of California San Diego La Jolla Institute for Cognitive Science. https://doi.org/10.21236/ADA164453

Selab, Essma and Ahmed Guessoum. 2015. Building TALAA, a free general and categorized Arabic corpus. In *ICAART (1)*, pages 284–291. https://doi.org/10.5220/0005352102840291

Sicilia-Garcia, Elvira I., Ji Ming, Francis J. Smith, et al. 2003. Extension of Zipf's law to word and character *n*-grams for English and Chinese. In *International Journal of Computational Linguistics & Chinese Language Processing, Volume 8, Number 1, February 2003: Special Issue on Word Formation and Chinese Language Processing*, pages 77–102.

Smith, Ray. 2011. Limits on the application of frequency-based language models to OCR. In *2011 International Conference on Document Analysis and Recognition*, pages 538–542. https://doi.org/10.1109/ICDAR.2011.114

Stolcke, Andreas. 2002. SRILM - an extensible language modeling toolkit. In *Seventh International Conference on Spoken Language Processing*. https://doi.org/10.21437/ICSLP.2002-303

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, 30:5998–6008.

Welsh, Dominic. 1988. *Codes and Cryptography*. Oxford University Press.

Werbos, Paul J. 1990. Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560. https://doi.org/10.1109/5.58337

Wikimedia Foundation. Wikimedia downloads.

Zhang, Hui and David Chiang. 2014. Kneser-Ney smoothing on expected counts. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 765–774. https://doi.org/10.3115/v1/P14-1072

Ziemski, Michał, Marcin Junczys-Dowmunt, and Bruno Pouliquen. 2016. The United Nations Parallel Corpus v1.0. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3530–3534.

Zipf, George Kingsley 2016. *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*. Ravenio Books.