# TEXT MINING AT MULTIPLE GRANULARITY: LEVERAGING SUBWORDS, WORDS, PHRASES, AND SENTENCES

BY

AHMED EL-KISHKY

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2020

Urbana, Illinois

Doctoral Committee:

Professor Jiawei Han, Chair
Professor ChengXiang Zhai
Professor Tarek Abdelzaher
Dr. Joy Zhang, Head of AirBnB AI Lab

## ABSTRACT

With the rapid digitization of information, large quantities of text-heavy data is being constantly generated in many languages and across domains such as web documents, research papers, business reviews, news, and social posts. As such, efficiently and effectively searching, organizing, and extracting meaningful information and data from these massive unstructured corpora is essential to laying the foundation for many downstream text mining and natural language processing (NLP) tasks.

Traditionally, NLP and text mining techniques are applied to the raw texts while treating individual words as the base semantic unit. However the assumption that individual word-tokens are the correct semantic granularity does not hold for many tasks and can lead to many problems and poor task performance. To address this, this work introduces techniques for identifying and utilizing text at different semantic granularity to solve a variety of text mining and NLP tasks. The general idea is to take a text object such as a document, and decompose it to many levels of semantic granularity such as sentences, phrases, words, or subword structures. Once the text in represented at different levels of semantic granularity, we demonstrate techniques that can leverage the properly encoded text to solve a variety of NLP tasks. Specifically, this study focuses on three levels of semantic granularity: (1) subword segmentation with an application to enriching word embeddings to address word sparsity (2) phrase mining with an application to phrase-based topic modeling and (3) leveraging sentence-level granularity for finding parallel cross-lingual data.

The first granularity we study is subword-level. We introduce a subword mining problem that aims to segment individual word tokens into smaller subword structures. The motivation is that, often, individual words are too coarse of a granularity and need to be supplemented by a finer semantic granularity. Operating on these fine-grained subwords addresses many important problems in NLP namely the long-tail data-sparsity problem whereby most words in a corpus are infrequent, and the more severe out-of-vocabulary problem. To effectively and efficiently mine these subword structures, we propose an unsupervised segmentation algorithm based off a novel objective: transition entropy. We use ground-truth segmentation to assess the quality of the segmented words and futher demonstrate the benefit of jointly leveraging words and subwords for distributed word representations.

The second granularity we study is phrase-level and the phrase mining task to transform raw unstructured text from a fine-grained sequence of words into a coarser-granularity sequence of single and multi-word phrases. The motivation is that, often, human language

contains idiomatic multi-word expressions and fine-grained words fail to capture the right semantic granularity; proper phrasal segmentation can capture this true appropriate semantic granularity. To address this problem, we propose an unsupervised phrase mining algorithm based on frequent significant contiguous text patterns. We use human-evaluation to assess the quality of the mined phrases and demonstrate the benefit of pre-mining phrases on a downstream topic-modeling task.

The third granularity we study is sentence-level granularity. We motivate the need for a sentence-level granularity for capturing more complex semantically complete spans of texts. We introduce several downstream tasks that leverage sentence representations in conjunction with finer-grained units in a cross-lingual text mining task. We experimentally show how leveraging sentence-level data for cross-lingual embeddings can be used to identify cross-lingual document pairs and parallel sentences – data necessary for training machine translation models.

*To my family and friends, for their love and support.*

## ACKNOWLEDGMENTS

I would like to extend my deepest gratitude to my advisor Professor Jiawei Han for his guidance and support during my Ph.D. This dissertation is possible due to his guidance, insights, critiques, and motivation. I also wish to thank other members of my Ph.D. committee, Prof. ChengXiang Zhai, Prof. Tarek Abdelzaher and Dr. Joy Zhang for their valuable suggestions with respect to my dissertation. I am also grateful to my collaborators and many other students and researchers who contribute to my and the Data Mining Group's research in many different ways. I would like to thank my collaborators in research Professor Xiang Ren at The University of Southern California, Dr. Chi Wang at Microsoft Research, Dr. Clare Voss at the Army Research Lab, Dr. Aston Zhang at Amazon, and Dr. Chenguang Wang at Amazon. I am also fortunate enough to have worked with and been a part of an amazing research group including many talented peers, students, and researchers. I would like to thank the members and alumni of The Data and Information Systems Laboratory including Chi Wang, Fangbo Tao, Chao Zhang, Yu Shi, Jonathan Kuck, Fang Guo, George Brova, Yanglei Song, Liyuan Liu, Yucheng Chen, Yihan Gao, Huan Gui, Quanquan Gu, Prof. Jing Gao, Shi Zhi, Jialu Liu, Jingjing Wang, Dr. Meng Jiang, Meng Qu, Stephen Macke, Doris Xin, Jingbo Shang, Jiaming Shen, Dr. Quan Yuan, Dr. Qi Li, Carl Yang, Xiaotao Gu, Yu Meng, Xiang Ren, Marina Danilevsky, Brandon Norick, Xuan Wang, Yuning Mao, Jinfeng Xiao, Qi Zhu, Jiaxin Huang, Xinyang Zhang, Yu Zhang, Sha Li, Wanzheng Zhu, Po-Wei Chan, Tobias Kin Hou Lei, Hongkun Yu, Min Li, and Qi Wang for their assistance, collaborations, and great memories we've shared. Finally, I would like to thank my parents for their understanding and encouragement during my pursuit of the Ph.D. degree.

## TABLE OF CONTENTS

# CHAPTER 1: INTRODUCTION

## 1.1 BACKGROUND AND MOTIVATION

With the exponential growth of multilingual unstructured text data in the form of web articles, social media posts, research publications, and business-generated data, automated methods of extracting information and inducing structure have become necessary to effectively and efficiently utilize and organize this trove of data. Text-mining and natural language processing techniques have been applied to unstructured text to perform downstream tasks such as machine translation, classification, information extraction, unsupervised clustering, as well as text-collection summarization, visualization, and exploration. To illustrate the variety of downstream text-mining tasks, let us consider the following examples:

**Example 1.1 (Topic Modeling)** *Clustering the documents and modeling the underlying topics is a useful step in organizing large text corpora. Consider an exploratory setting on a large archival unit. When reading or exploring a document over a topic, the reader can be suggested documents with similar topics. Additionally, clustering documents and visualizing their underlying topical distribution allows for automatic summarization without directly reading the vast amount of unstructured data.*

**Example 1.2 (Word Embeddings)** *Word embeddings have become a valuable tool in text-mining applications as they provide a low-dimensional representation that captures semantic relationships between words. Leveraging the large amounts of unsupervised text corpora as training data, word embeddings facilitate learning text-mining and NLP models to be developed with relatively small amounts of training data. However, when training word embeddings, data sparsity issues like infrequent words can result in poor word embeddings. Additionally, during inference time, word-level embeddings fail to provide adequate and meaningful representations for out-of-vocabulary words.*

**Example 1.3 (Machine Translation & Cross Lingual Mining)** *With rapid globalization, performing NLP tasks such as machine translation for a variety of languages has become crucial. Yet the availability of parallel data for many low-resource languages makes it difficult. As such, the development of language-agnostic techniques to automatically sift through large quantities of web-data in search of training data is crucial to inter-nationalization of NLP*

### 1.1.1 Leveraging Multiple Semantic Granularity for Text Mining Problems

Most NLP and text-mining tasks treat text as sequences of word tokens with individual words serving as the smallest semantically-meaningful unit. While this assumption often holds for a majority of cases and yields acceptable results in downstream tasks, there are many cases where word-level granularity is insufficient. In reality, natural language has variable-size semantic units.

For example, in English and other languages, there are a variety of multi-word idiomatic expressions that are *non-compositional* such as "piece of cake" or "a dime a dozen" where the intended semantic meaning is not derivable from constituent words. In addition phrases in general are often more human-interpretable and informative than their underlying unigrams.

This phenomenon has been previously explored in many fields including information retrieval. For example, previous works have identified that infrequent unigram terms are more informative than frequent ones for retrieval as frequent in a corpus are less discriminative and informative. In these situations, previous works combine unigrams to form more discriminative ngrams for better retrieval [1]. The success of these previous works support the idea operating on the proper level of semantic granularity can improve the downstream task.

We introduce an additional example on the other end of the spectrum whereby many words in text corpora are infrequent with about 50% appearing only once [2]. Naturally this translates to many out-of-vocabulary word in test settings. Despite this, human can often infer the meaning of these infrequent words based on their morphology. This motivates leveraging the semantic information in subwords to infer meaning in infrequent words. More generally, when the base-semantic unit is too coarse due to data sparsity, a finer granularity structure is necessary.

Drawing on these two examples whereby both coarser-grained and finer-grained semantic units can improve text-mining tasks, we investigate methods for identifying and leveraging multiple levels of semantic granularity for a variety of text-mining and NLP tasks.

**Why is this Task Challenging?** Our goal is to take as input large text corpora and automatically segment each corpus into semantically-meaningful text segments at multiple granularity. More specifically, we take raw text composed of sequences of word tokens and automatically segment them into sentences, single and multi-word phrases as well as identifying semantically meaningful subword structures. This task is not trivial due to the following reasons:

1. **Lack of training data.** Annotated training data is scarce for identifying segments such as phrases and subwords. Additionally, such methods don't generalize to new

languages and scripts. As such unsupervised and weakly-supervised techniques are necessary.

2. **Domain-Specific Corpora** Many important text-mining applications are necessary on domain-specific corpora such as scientific literature, social media data, and news articles. Working with these domain-specific corpora involves operating on data that may not be intuitive to evaluate. Techniques proposed should be easy to generalize to these domain-specific corpora.

3. **Language Agnosticism.** While most NLP is focused on English, with the growth of the World Wide Web, large quantities of text data is being generated in many languages and many of these are low-resource languages which would benefit even more from identifying the correct semantic granularity. As such, techniques should generalize to more languages than English.

Additionally, once corpora have been segmented into the proper granularity for a task, effectively leveraging these segments to solve a task requires crafting new techniques to effectively utilize text input at multiple granularity.

## 1.2 DISSERTATION STATEMENT

We study various tasks in natural language processing and text mining, and propose novel solutions to them. The proposed solutions introduce techniques for segmenting raw text into spans at multiple semantic granularity and integrate these segments to solve each mining task. Specifically, this dissertation offers evidence for the following statement:

**Dissertation Statement:** Segmenting raw text into multiple levels of semantic granularity is a crucial first step in many NLP and text mining tasks. Leveraging multiple levels of granularity over a single segmentation can improve a variety of downstream text mining tasks such as topic modeling, learning word embeddings, and mining parallel texts.

## 1.3 DISSERTATION CONTRIBUTIONS

To support the dissertation statement in Section 1.2, we study and make contributions to following problems.

### 1.3.1 Subword Mining & Subword Segmentation

Our first problem aims to automatically identify fine-grained morphemes or subword-level semantic units. The proposed method applies novel unsupervised and supervised segmentation techniques to split words into morphemes. Results show that our algorithm successfully identifies subwords on ground-truth annotated datasets across multiple languages.

**Downstream Task: Word Embeddings**
To demonstrate the utility of our word-segmentations in data-sparsity-sensitive tasks, we incorporate our mined morphemes to enrich a popular subword-based embedding algorithm. We modify FastText to incorporate our segmented subwords to enrich the embedding process. We evaluate against baseline subword segmentation algorithms, and demonstrate superior word embeddings as measured by superior performance on analogies, word similarity, and a downstream language modeling task.

### 1.3.2 Phrase Mining & Phrasal Segmentation

Our second problem aims at leveraging large-quantities of raw text data for unsupervised phrasal segmentation. That is given a text corpus, segment the corpus from a sequence of word tokens to a sequence of single and multi-word phrases. Our algorithm utilizes a significance-score approach to guide a novel bottom-up agglomerative merging algorithm.

**Downstream Task: Topic Modeling**
To demonstrate the utility of phrases vs unigrams, we incorporate our phrase-segmented corpus in a downstream topic modeling task. We propose a novel variation of the popular topic-modeling algorithm LDA that principally leverages single and multi-word phrases. Our proposed PhraseLDA not-only generates more human-interpretable topics, but also demonstrates better held-out perplexity in relation to vanilla LDA. This suggests that topic modeling benefits from leveraging input text both at the word and phrase-level.

### 1.3.3 Leveraging Sentence Representations for Text Similarity

Our third problem aims at leveraging sentence representations to evaluate semantic text similarity between segments in different languages. We demonstrate how cross-lingual sentence embeddings can be used to perform not only cross-lingual sentence similarity, but also cross-lingual document similarity.

**Downstream Task: Mining Cross-lingual Parallel Data**
To demonstrate the utility of cross-lingual sentence representations we introduce techniques for mining cross-lingual parallel documents. These documents have a variety of utility from training zero-shot cross-lingual document classification, to mining parallel sentences for machine translation training data. We demonstrate a sentence-representation approach for aligning parallel sentences and the benefit this affords for training low-resource machine translation models.

## 1.4 ORGANIZATION OF THE DISSERTATION

The remainder of this thesis is organized as follows. In Chapters 2 and 3, we present MorphMine [3] and Constrained Seq2Seq [4], an unsupervised and supervised solution to the subword segmentation problem respectively. Each chapter discusses a subword mining and segmentation algorithm as well as a subword-based word embedding technique. In Chapter 4, we focus on the phrase mining and phrasal segmentation problem and introduce a way of solving it. This chapter discusses ToPMine, a phrase mining and segmentation step followed by a downstream phrase-based topic modeling that uses our phrasal-segmentation [5]. In Chapter 5 we construct a large comprehensive dataset of cross-lingual document pairs in many languages and demonstrate a simple technique for mining parallel sentences [6]. We demonstrate how cross-lingual sentence representations can mine valuable training data for machine translation systems. In Chapter 6, we further extend this work and demonstrate how cross-lingual sentence representations can be used to identify additional cross-lingual document pairs [7]. Finally, in Chapter 7 we analyze the relationships between the different levels of semantic granularity and conclude by discussing future works.

# CHAPTER 2: UNSUPERVISED SUBWORD MINING FOR ENRICHING WORD REPRESENTATIONS

In this chapter, we investigate the motivation for decomposing words into subword structures such as morphemes. Recognizing the need for language-agnostic subword segmentation, we propose an unsupervised technique for segmenting words into human-verified morphemes. We demonstrate the utility of these morphemes by utilizing them to enrich distributed word embeddings. Experimental results support the claim that utilizing word level information in conjunction with subwords outperforms the use of each independently.

## 2.1 INTRODUCTION

Decomposing individual words into finer-granularity morphemes is a necessary step for automatically preprocessing concatenative vocabularies where the number of unique word forms is very large. While linguistic approaches can be used to tackle such segmentation, such rule-based approaches are often tailored to specific languages or domains. As such, data-driven, unsupervised methods that forgo linguistic knowledge have been studied [8, 9]. Typically, these methods focus on segmenting words by applying a probabilistic model or compression algorithms to a full text corpus. The resultant morphemes from these methods have been primarily shown to improve neural machine translation [10, 11].

One natural application to utilize these semantically meaningful morphemes is distributed word representation. There are many advantages to using distributed continuous word representations as an alternative to one-hot bag of words [12, 13] since this leads to a dimensionality much smaller than the vocabulary size of a corpus. It has been shown that working with low-dimensional representations not only demonstrates computational efficiency, but also captures syntactic and semantic regularities while boosting the performance in text classification, sequential classification, sentiment analysis, and machine translation [14, 15, 16, 17, 18]. As such, many methods have been developed to learn these word representations from large, unlabeled text corpora [19, 20, 21].

Despite many advances, unsupervised learning of distributed representations can struggle in learning adequate vectors for infrequent words. This problem is ubiquitous because most text corpora demonstrate long-tail distributions in relation to word frequency, with often $40\% - 60\%$ of words in a vocabulary appearing just once in a corpus [2]. Naturally, many methods fail to produce meaningful embeddings for unseen (out-of-vocabulary) words. Using morphemes for parameter sharing not only bolster training data for infrequent words but also allow for constructing meaningful word embeddings for unseen words.

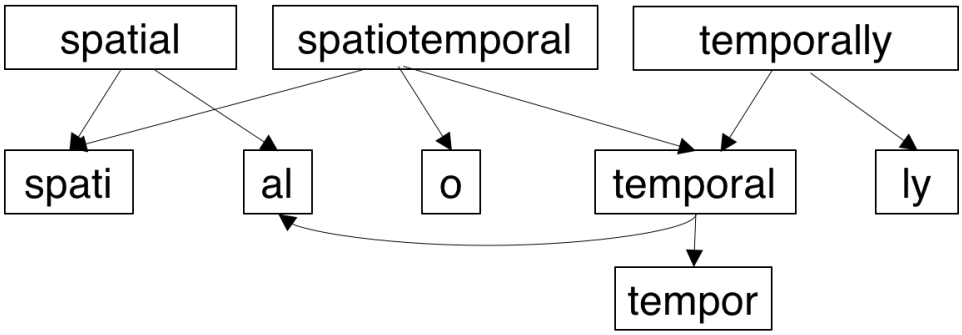



**Figure 2.1:** Hierarchical segmentation of words.

What differentiates our method from others is *extracting morphemes at multiple granularity*. As seen in Figure 2.1, morphologically-rich words share semantically-meaningful morphemes. Larger morphemes carry more semantic meaning, but are often infrequent within the vocabulary and discarded in favor of more frequent finer-grained morphemes in other methods. Yet, including both fine and coarse-grained morphemes, can better semantically tie the meanings of words that share them. With this motivation, we propose MorphMine, a continuation on preliminary work [22]. We formalize the novel methodology by framing the morpheme segmentation as entropy-boundary identification and segmentation with a parsimony criterion. We introduce a global resegmentation to refine and improve the segmentation after the initial segmentation. Finally, we evaluate our method on a variety of datasets and tasks in multiple language and demonstrate how multi-granular morphemes can be used for enriching word embeddings for robustness to data-sparsity.

## 2.2 RELATED WORK

In morphological analysis, predictability has been suggested for detecting morpheme structure. An early quantitative metric proposed was the number of different variations of morphemes following a morpheme sequence whereby a high number of variations indicates a morpheme boundary [23]. While this work provided influential insight into useful metrics for morpheme-detection, the main objective was developing a scoring function for identifying candidate morphemes, not segmentation. Following this line of work, were methods to identify frequent morphemes and affixes [24, 25, 26]. These methods identify a high-precision but low-recall subset of morphemes. Similarity measures have been proposed for detecting affixes by comparing words and identifying similar and dissimilar parts. These methods utilize a variety of techniques including edge-alignment, adding words and their reverse to

tries [27, 28]. Unfortunately, these methods can only identify prefix and suffix morphemes, neglecting morphemes that occur between other morphemes. One model segments words by applying the minimum description length principle to minimize the vocabulary while maintaining the likelihood of the corpus data [9, 29]. Other fixed-vocabulary methods apply a unigram language model approach to identifying morphemes (also called wordpieces) and has been successfully applied to a variety of NLP tasks [30, 31]. Similarly, the byte-pair compression algorithm has been used to identify morphemes for neural machine translation tasks [10].

To address data-sparsity when learning word embeddings, some methods apply a factored neural language model where words are represented as a set of features including morpheme information [32]. Other methods add morphological similarity features into a neural network along with the context features [33, 34]. Other methods take morphologically annotated data and train log-bilinear models to jointly predict context words and morphological tags [35]. The method we utilize for our embeddings is FastText [36]. While FastText utilizes all the possible character $n$-grams up to certain length for enrichment, we only utilize high-quality morphemes in MorphMine. Finally, many methods have utilized characters as the base unit for embedding. Some approaches treat each word as a sequence of characters and apply RNNs or convolutional networks [37, 38, 39].

## 2.3 PRELIMINARIES

The input is a corpus $W$, consisting of $|W|$ words: $W = w_1, \ldots, w_{|W|}$. From this corpus, we construct a vocabulary of unique words, $V$, of size $|V|$ such that $\forall w \in W, w \in V$. In addition, the $v^{th}$ word is a sequence of $|v|$ characters: $c_{v,i}, i = 1, \ldots, |v|$. For convenience we index all the unique characters that compose the input vocabulary with $C$ characters and $c_{v,i} = x, where\ x \in \{1, \ldots, C\}$ means that the $i^{th}$ character in $v^{th}$ word is the $x^{th}$ character in the character vocabulary.

Given an input corpus consisting of a word sequence and a vocabulary list of unique words, our goal is to segment the vocabulary list to identify human-interpretable and semantically meaningful morphemes, then utilize these morphemes for parameter sharing when learning distributed word representations from the corpus.

**Definition 2.1 (Morpheme Formalization)** *A* morpheme *is a sequence of characters:* $m = \{c_{v,i}, ..., c_{v,i+n}\}$ *where* $n > 0$

- *A* partition *over vocabulary word* $v$ *is a sequence of morphemes:* $\mathcal{G}_v = (m_{v,1}, \ldots, m_{v,G_v})$ *where* $G_v \geq 1$ *s.t. the concatenation of the morphemes is the original word.*

In Definition 2.1 we formalize a morpheme and the resultant partition from segmenting a word into morphemes. In addition we outline the desired properties of the framework as follows:

1. extracts semantically meaningful, human-interpretable at multiple granularity
2. the method is general and applies to words on a variety of languages
3. enriching word morphemes improves word embeddings
4. the overall method is computationally efficient

### 2.3.1 The MorphMine Framework

At a high-level, our proposed framework can be summarized into two sequential steps: (1) mining candidate morpheme patterns and character co-occurence statistics, and (2) performing word segmentation into finer-grained morphemes. In step one, by applying an information-theoretic metric to detect candidate morpheme boundaries, we identify candidate morphemes within each vocabulary word. These morphemes are propagated to other words and pruned to ensure high-quality. For step two, from this candidate pool, we then apply an unsupervised dynamic programming segmentation algorithm to select a subset of these morphemes that best segment each word. Segmentation and partition induction further prune away low-quality morpheme candidates leaving a high-quality morpheme vocabulary. After inducing a partition on each word, we can recursively segment each morpheme to finer granularity. Applying this two-step process maps each word in the input vocabulary to a set of high-quality morphemes. The resultant morphemes from the hierarchical segmentation can then be used for downstream NLP and text analysis tasks.

The main objective in morpheme pattern mining is to collect aggregate statistics on morpheme patterns that can be used to score and reason about the quality of candidate morphemes. These statistics are then used in the word segmentation algorithm. For each character $n$-gram that appears more than once in the vocabulary, there is a potential for parameter sharing via the candidate morpheme as it appears in multiple vocabulary words. Additionally the frequency counts of these morphemes will be used for entropy-boundary computation to identify and score potential morpheme candidates. These candidates are input to the word-segmentation algorithm that attempts to apply Occam's Razor by positing that using the fewest morphemes in the segmentation best segments each word [40]. This process is then applied recursively to each morpheme to obtain finer-grained morphemes.

By inducing a partition over each vocabulary word, we effectively transform each word into a bag-of-morphemes. These morphemes can be shared among other words within the vocabulary and model the belief that words that share morphemes, share semantic meaning. This is done by individually embedding each morpheme; these morpheme embeddings are then combined to form the final word embedding. Because a word embedding is constructed from the embeddings of constituent morphemes, words that share constituent morphemes will be *partially* constructed from similar morpheme embeddings.

We expound upon our morpheme-mining algorithm and its evaluation in a popular embedding framework in Section 2.4.

## 2.4 METHODOLOGY

Given an input vocabulary list $V$, MorphMine segments each word into non-overlapping character $n$-grams (morphemes). Our method is non-parametric, hierarchical and data-driven, allowing for good cross-domain performance without incorporating domain-specific knowledge or linguistic rulesets. The entire morpheme segmentation can be performed as an easy preprocessing step to the vocabulary for downstream text-related tasks. To learn a morpheme vocabulary and segment an input vocabulary, MorphMine performs the following steps: (1) mine morpheme pattern counts and compute entropy statistics, (2) apply parsimonious segmentation to identify the best *locally-consistent* segmentation, (3) recompute morpheme counts after segmentation to ensure *global-consistency* and maximize parameter sharing of morphemes, and (4) re-segment using refined morpheme vocabulary counts.

We apply an entropy-based scoring function to identify morpheme boundaries: generating candidate morpheme vocabulary. Given this collection of morphemes and their counts, the next step is to apply a dynamic-programming algorithm to segment each word into high-quality morphemes. For each word, the parsimonious segmentation identifies the *most-likely* segmentation using the *fewest number of morphemes*. This step discards a large number of lower quality candidates morphemes from our vocabulary and allows for a more-accurate estimate of morpheme counts. Using the refined vocabulary, we can then re-segment and improve the overall quality of segmentation. The re-segmentation biases towards selecting locally-consistent segmentations that globally-optimize for morpheme parameter sharing. That is, the resegmentation favors morphemes used in the segmentation of other words in the vocabulary. Finally, the resultant collection of morphemes for each word can be utilized to enrich word embeddings.

### 2.4.1 Morpheme Vocabulary Generation

Our segmentation of words into morphemes relies on the idea of morpheme compositionality. That is, the input vocabulary can be constructed by composing morphemes drawn from a smaller morpheme vocabulary. As such we introduce an approach for creating the initial morpheme vocabulary: prefix, suffix, and root-word candidates.

Prefix & Suffix Generation.

We posit that prefixes and suffixes can be identified through the concept of transition predictability.

**Definition 2.2 (Transition Predictability)** *Transition predictability is a quantification of being able to predict the next character in a word given a prefix.*

Previous works have attempted to quantify Definition 2.2, by using number of character choices following a prefix in a vocabulary [23, 24, 25, 26]. For example, many words begin with the prefix, "*pre*" such as, *prepaid*, *preview*, *presoak*, etc. Given the large number of words with the prefix "pre", the transition from "pr" to "pre" predictable, but "pre" to a longer prefix is not as predictable as many words have "pre" followed by a variety of root words.

Unfortunately, using raw counts to identify high-unpredictability boundaries for prefixes does not generalize to large vocabularies and different languages as the character count is arbitrary. As such we propose a metric on the normalized distribution of character choices: information entropy [41]. Let $v$ be a word consisting of $|v|$ characters and $m_i$ be a prefix of $v$ ending at the $i_{th}$ character of $v$. For each candidate prefix boundary $i$ for $i \in [1 \dots |v|]$, the prefix transition unpredictability can be quantified with information entropy. As the transition between a prefix and longer prefixes can be modeled as a multinomial of support size $C$, the character vocabulary, we use the multinomial distribution entropy:

$$\begin{aligned} H(X) &= -\log(n!) - n\sum_{j=1}^{C} p_j \log(p_j) + \\ &\quad \sum_{j=1}^{C}\sum_{x_j=0}^{n} \binom{n}{x_j} p_j^{x_j}(1-p_j)^{n-x_j}\log(x_j!) \\ &= -\sum_{j=1}^{C} p_j \log(p_j), \text{ when n=1} \end{aligned} \tag{2.1}$$

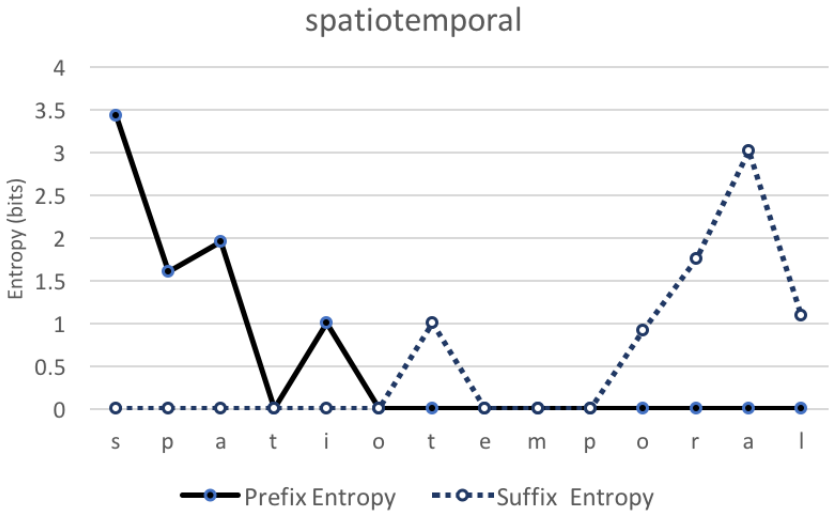



**Figure 2.2:** Prefix and suffix transition entropy.

This is the entropy of a multinomial over support $C$ for $n$ independent trials each of which leads to a success for exactly one of the $C$ characters. Because we consider a single trial, $n = 1$, we simplify it into entropy of the categorical distribution. For the prefix $m_i$:

$$H(m_i) = -\sum_{j=1}^{C} \mathsf{P}(m_i \oplus c_j | m_i) \times \log_2 \mathsf{P}(m_i \oplus c_j | m_i), \tag{2.2}$$

where $\oplus$ denotes the binary string concatenation of two strings and the transitional prefix probability is estimated as:

$$\mathsf{P}(m_i \oplus c_j | m_i) = \frac{f(m_i \oplus c_j)}{f(m_i)} \tag{2.3}$$

and $f(m_i)$ denotes the frequency of a prefix $m_i$ in the input vocabulary list. The entropy of suffixes can, without loss of generality, be similarly computed by reversing each word in the vocabulary and treating each suffix as a prefix.

The information entropy of each possible prefix and suffix in the vocabulary is computed in *linear time* with relation to unique vocabulary size using a prefix tree data structure to store counts over prefixes. Given entropy scores for each prefix and suffix, scores are computed for each candidate split point in each word. Under the entropy scoring of prefixes and suffixes, we identify *local maxima* in entropy as candidate boundaries for prefixes and suffixes. That is entropy of a prefix one-character shorter and one-character longer should be lower than a candidate prefix boundary. This is intuitive as under our principle of compositionality

assumption, complex words are formed by concatenating morpheme structures. As such, given an incomplete morpheme, the next character can easily be predicted, but given a complete morpheme, any number of new morphemes can be concatenated to the completed morpheme increasing the unpredictability and thus entropy. These high-entropy positions thus serve as a strong indicator of morpheme boundaries. As seen in Figure 2.2, for the word "spatiotemporal", candidate prefixes and suffixes are found at boundaries exhibit a local maxima in entropy. For "spatiotemporal", candidate prefixes are "spa" and "spati" while candidate suffixes include "al" and "temporal".

Root Word Generation

Utilizing entropy-scoring, it is possible to detect morpheme structures that occur at the beginning or end of a word. However, many words often contain morpheme structure between prefixes and suffixes. For each prefix and suffix candidate identified in a word, it is possible to generate many candidate root words by stemming the word and removing prefixes and suffixes. This creates a high-quality pool of root words to be used in conjunction with prefixes and suffixes for segmenting the vocabulary.

**Example 2.1 (Root Extraction)** *Removing prefixes and suffixes yields candidate roots.*

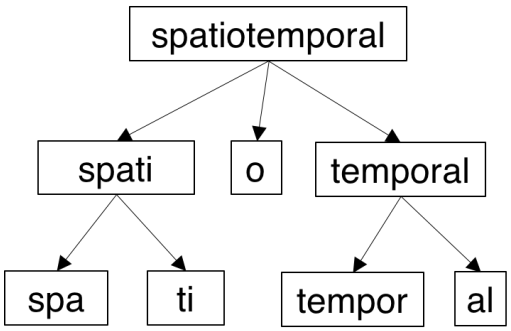
***[pre]*** + *authenticat* + ***[ion]***
***[pre]*** + *authentication*

*The characters grouped together by [] are prefixes and suffixes. When removed, the remaining underlined character-sequence represent candidate root words.*

As seen in Example 2.1, when stripping the combinations of prefixes and suffixes of a word, the remaining character sequence is considered a candidate root word. We apply some filtering conditions for each candidate root to test the viability as a shareable root. These include: (1) a minimum support of two within the vocabulary, and (2) the minimum root length of four. Additionally, for each word in the vocabulary, after stripping prefixes and suffixes, the candidate root words that meet the constraints are added to the morpheme vocabulary.

### 2.4.2 Parsimonious Morpheme Segmentation

After generating a morpheme vocabulary using entropy-based predictability metric for boundary detection, we segment words into morphemes, utilizing this morpheme vocabulary.




(a) Parsimonious Segmentation

| Candidate Subwords |
|---|
| spa |
| spati |
| temp |
| tempor |
| temporal |
| al |

(b) Candidate Morphemes

**Figure 2.3:** Segmentation of the word "spatiotemporal" using disjoint interval covering.

The algorithm first identifies candidate morphemes from the morpheme vocabulary within a word, then selects a subset of these candidate morphemes that best segment the word. The main insight is a per-word implementation of Occam's Razor. That is, according to the preference for parsimonious hypotheses, we posit that each word is composed of the *fewest number of morphemes that maximally cover the word.*

As seen in Figure 2.3, morphemes present in the target word are identified and recursive segmentation is performed to segment the word into morphemes. Example 2.2 demonstrates how the candidates are used to segment the target word under the *parsimony criterion.*

**Example 2.2 (Parsimonious Segmentation)** *Segmentations are scored based on word coverage and the number of morphemes.*

| ***Segmentation*** | ***# Morpheme*** | ***Coverage*** |
|---|---|---|
| [**spa**] + tio+ [**temporal**] | 2 | 11 |
| [**spati**] + o + [**temporal**] | 2 | 13 |
| [**spati**] + o + [**tempor**] + [**al**] | 3 | 13 |
| [**spa**] + tio [**tempor**] + [**al**] | 3 | 11 |

**Table 2.1:** Candidate segmentations of a word. The highlighted row displays the maximally parsimonious morpheme segmentation.

Subsets of non-overlapping candidate morphemes are used in segmentation, and the most parsimonious segmentation is selected. Because the possible subsets of candidate morphemes form a power set, direct enumeration of each segmentation quickly proves computationally slow for even a modest number of candidate morphemes. To identify the most parsimonious

segmentation, we abstract our parsimonious morpheme segmentation task into a general problem we dub *Disjoint Interval Covering* and demonstrate that this problem can be solved via dynamic programming in linear time. We formalize the disjoint interval covering problem as follows:

**Definition 2.3 (Disjoint Interval Covering)** *Given an input $N \in \mathbb{N}$ and a set $A$ of pairs $(a, b) : a, b \in \{1 \dots N\} \times \{1 \dots N\}$ and $a < b$, find the smallest subset $B \subseteq A$ such that $|\bigcup_{x \in B} x|$ is maximized, $|B|$ is minimized, and $\forall x, y \in B : x \neq y \Rightarrow x \cap y = \varnothing$.*

As seen in Definition 2.3, the input is a set of pairs $A$ and a positive integer $N$. Within the segmentation perspective, these refer to position index boundary pairs for candidate morphemes and the word length. Given these inputs, the objective is to select a minimum subset of disjoint morphemes that maximally cover the word. That is, select a set of disjoint morpheme whose combined length is as close as possible to the word length.

$$F(j) = \max_{0} \min_{1} \left\{ \begin{array}{ll} (0,0), & j < 1 \\ F(j{-}1), & j \geq 1 \\ \max\limits_{0} \min\limits_{1} \{F(i{-}1)_0 + (j{-}i{+}1), F(i{-}1)_1{+}1\}, & j \geq 1 \\ \qquad\qquad\quad (i,j) \in A & \end{array} \right\} \tag{2.4}$$

We define a recurrence to the disjoint interval covering problem in Equation 2.4. This recurrence posits that the segmentation that maximally covers the word is either the solution for the current word minus the ending character, or the max-covering, min-morpheme solution utilizing all morphemes that have a right boundary index equal to the index of the end of the word. With proper memoization, it is evident that for a word of size $|v|$, there are $|v|$ subproblems to solve. In addition, because each interval's right boundary corresponds to the word size, each interval is iterated over a constant number of times. As such, for word $v$, the total, memoized complexity of this segmentation is $\mathcal{O}(v + |A_v|)$ where $A_v$ indicates the pre-segmentation morphemes that are substrings of word $v$, making our overall framework of linear complexity – $\mathcal{O}(V)$.

Algorithm 2.1 presents the morpheme segmentation algorithm. The algorithm takes as input a word and a collection of intervals corresponding to index boundaries of candidate morphemes within the word. It then proceeds to select a set of intervals that maximally cover the word while utilizing the fewest number of intervals. Solutions to subproblems are memoized as to avoid repeated computation. While the algorithm returns a memoization list of best segmentations that terminate at each index, proper backstracking can construct all

**Algorithm 2.1:** DP Parsimonious Segmentation (DP)

**Input:** Word $v$, morpheme Intervals $A_v$
**Output:** Optimal segmentation $S$

```
n[0] ← 0; c[0] ← 0; p[0] ← null;
for j := 1 to N_v do
    num ← n[j-1]; cov←c[j-1]; pair ← p[j-1];
    for (i, j) ∈ A_v do
        cov′ ← c[i-1]+(j-i+1)
        num′ ← n[i−1]+1
        if cov′ > cov then
            cov ← cov′; num ← num′;
            pair ← (i, j);
        end
        if cov′=cov ∧ num′<num then
            num ← num′;  pair ← (i, j)
        end
    end
    n[j] ← num;  c[j] ← cov;  p[j] ← pair;
end
return p
```

possible parsimonious segmentations. In the next subsection we demonstrate how to select among equally-parsimonious segmentations.

Maximum Likelihood Scoring

While Algorithm 2.1 identifies the most parsimonious segmentation, the algorithm often returns many segmentations with equal parsimony. As such, after applying Algorithm 2.1, maximum likelihood is used to select *the most likely segmentation* among these candidate segmentations.

Given the previous counts of candidate morphemes obtained, it is simple to compute the most likely segmentation among the candidate set of parsimonious segmentations given an independence assumption. Given a segmentation (partition of morphemes) over word $v$, $\mathcal{G}_v$, one can calculate the likelihood over the partition:

$$\mathcal{L}(\mathcal{G}_v) = \prod_{m \in \mathcal{G}_v} \mathsf{P}(m) \propto \prod_{m \in \mathcal{G}_v} f(m) \tag{2.5}$$

The independence assumption yields and discarding the normalization yields a simple product over each morpheme count $f(m)$ in the partition. This follows as all parsimonious

partitions have the same number of morphemes and as such, the normalization constants for the probabilities should be the same for all parsimonious segments. One additional important constraint we place on our most-likely partition is that, during training and learning of the morpheme vocabulary, the most-likely partition cannot have any morphemes that occur only once in the vocabulary. That is: $\forall m \in \mathcal{G}_v : f(m) > 1$. This ensures that each learned morpheme is shared at least with another word. As seen in Example 2.3, the largest product of counts is selected as the best segmentation. By applying the restriction that all morphemes must be shared at least once, MorphMine filters poor segmentations such as "*incompletenes* + *s*" where the morpheme "*incompletenes*" only appears once in the vocabulary.

**Example 2.3 (Most-likely Segmentation)** *Most likely segmentation from candidates.*

| ***Segmentation*** | $count(m_1)$ | $count(m_2)$ | *Likelihood Score* |
|---|---|---|---|
| [**incompletenes**] + [**s**] | 1 | 2072 | 2072 |
| [**incomplete**] + [**ness**] | 4 | 115 | 660 |
| [**in**] + [**completeness**] | 659 | 4 | 2636 |
| [**incomp**] + [**leteness**] | 4 | 2 | 8 |

**Table 2.2:** Candidate segmentation and morpheme counts. The highlighted row displays the most likely segmentation.

The mostly likely segmentation "*in* + *completeness*" is selected and in further steps, "completness" will be recursively decomposed into smaller morphemes "complet" and "ness" parsimoniously.

### 2.4.3 Local Segmentation.

Subsection 2.4.1 introduced the concept of utilizing high-entropy boundaries to create a morpheme vocabulary, and Subsection 2.4.2 introduced an algorithm for segmenting words into morphemes based on the principle of parsimonious disjoint interval covering and tie-breaking with maximum likelihood. In this subsection we demonstrate a high-level overview on how to apply these two methods to hierarchically segment words into multi-granular morphemes.

Following the steps from Subsection 2.4.1, an initial morpheme vocabulary is created. Within the vocabulary, we differentiate between prefixes, suffixes, and root words. As seen in Algorithm 2.2, Line 2, each morpheme found in the input word is mapped to an interval indicating its boundary indices within the word with the condition that prefix intervals must

**Algorithm 2.2:** Segmentation Algorithm (SEGMENT)

**Input:** Word $v$, morpheme Vocabulary SW
**Output:** Set of morphemes of $v$

```
output ← {v}
A_v ← {(i, j) for v_i ... v_j ∈ SW and j-i ≠ |v|}
if A_v = ∅ then
    return output
end
segmented ← DP(w, A_v)
for morpheme ∈ segmented do
    output ∪ SEGMENT(morpheme, SW)
end
return output
```

start at the beginning of the word, suffix intervals must terminate at the end of the word, and root word intervals can be located at any position within the word. In addition, the complete word is not included (to ensure the word segments to smaller morphemes). The algorithm terminates if the word cannot be further segmented. Otherwise, the word is segmented with the dynamic programming parsimonious segmentation algorithm. Each morpheme is then treated as a word and recursively segmented; the collection of all morphemes from segmentation are output.

### 2.4.4 Global Resegmentation

The parsimonious segmentation selects the most-likely locally-consistent segmentation of a word. Yet because each word is segmented independently, a morpheme that is present in two different words may not be selected because parsimonious segmentation is performed on both words independently. To address this, after performing one segmentation we utilize the resultant segmentation to refine the morpheme counts and prune infrequent morphemes from the morpheme vocabulary. Using this refined morpheme vocabulary and a more accurate morpheme count estimation, each word is re-segmented. The resultant segmentation is not only performed with a smaller morpheme vocabulary, but also favors the morphemes that other words have selected in their own parsimonious segmentations, creating a global consistency for the overall vocabulary segmentation.

**Example 2.4 (Re-segmentation with refined counts.)** *After one pass through the vocabulary and segmenting with parsimonious segmentation. Morpheme counts are re-computed using the resultant segmentations.*

| ***Segmentation*** | ***Counts*$_1$** | ***Max-Likelihood*$_1$** | ***Counts*$_2$** | ***Max-Likelihood*$_2$** |
|---|---|---|---|---|
| [**bit**] + [**emporal**] | 5,6 | **30** | 3,1 | 3 |
| [**bi**] + [**temporal**] | 4,6 | 24 | 3,5 | **15** |

**Table 2.3:** Candidate segmentation of a word. Highlighted row is the most likely segmentation using refined counts

*While initial counts and scores (Counts$_1$, ML$_1$) determine the locally optimal segmentation. After recomputing the morphemes post initial segmentation, refined counts and scores are computed (Counts$_2$, ML$_2$). The white row displays the initial best segmentation, while the grey row shows the best segmentation after refining morpheme counts.*

As seen in Example 2.4, in the first segmentation, the locally-consistent parsimonious segmentation favors the incorrectly segmented "*bit + emporal*" with likelihood 30 over "*bi + temporal*" with likelihood 24 as the likelihood is higher for the former. After one round of segmentation, it is apparent that the morpheme "*emporal*" was only selected once out of the possible six occurrences, while "*temporal*" was selected five out of six word segmentations. Resegmentation with these refined counts helps choose the correct segmentation "*bi + temporal*" with likelihood score 15 over 3. With this refined segmentation, these morphemes can be used in the morpheme-enriched word embedding learning.

### 2.4.5 Morpheme-Enriched Word Embedding

To efficiently utilize our mined morphemes to improve upon word embeddings, we modify the FastText model for word embeddings to use our extracted morphemes [36] to enrich *infrequent or out-of-vocabulary* words. As explained in the FastText paper, it is often the longest subword that captures the most semantic meaning. As such, we take each and every node in our word segmentation representing morphemes at every granularity and directly input the morphemes extracted from this layer to enrich each word in the vocabulary.

We begin with a brief review of FastText, and then demonstrate integrating morphemes in place of the standard FastText enumerated subwords. First, we note that FastText utilizes the skip-gram objective with negative sampling yielding the following objective (for simplicity, $\ell(x) = \log(1 + \exp(-x))$):

$$\sum_{x=1}^{W} \Big[ \sum_{c \in \mathcal{C}_x} \ell(s(w_x, w_c)) + \sum_{t \in \mathcal{N}_{x,c}} \ell(-s(w_x, t)) \Big], \tag{2.6}$$

where $w_x$ is the $x^{th}$ word in the corpus, $\mathcal{C}_x$ denotes the set of context words within a window of word $w_x$, and $\mathcal{N}_{x,c}$ denotes the set of negative examples sampled from the vocabulary. The scoring function is then adapted to incorporate morpheme information as $s(w_x, w_c) = \sum_{m \in w_x} \mathbf{z}_m^\intercal \mathbf{v}_c$ where each $\mathbf{z}_m$ denotes a morpheme embedding vector and the scoring function is a summation over morpheme embedding vectors in a dot-product with the context word vector. While FastText incorporates all contiguous substrings of lengths three to seven as morphemes in the scoring function, we posit that many of these morphemes are semantically not meaningful and, as such, degrade the overall quality of the learned embeddings. We claim that directly incorporating meaningful morphemes extracted by MorphMine for each word and summing over each morpheme's embedding results in higher quality distributed representations.

## 2.5 EXPERIMENTAL RESULTS

We introduce the datasets used and methods for comparison. We then evaluate our method on a morpheme segmentation task, a variety of embedding tasks, and a downstream language modeling task.

**Datasets**

- **English, German, and Turkish Vocabularies and Segmentations**. This dataset consists of three vocabulary lists in English, German, and Turkish with 156K, 290K and 90K unique vocabulary words, respectively. Each list is accompanied by approximately 1500 ground-truth segmentations consisting of a vocabulary word and its segmentation into constituent morphemes. These ground-truth segmentations were annotated as part of the MorphoChallenge [42].

- **English, German, and Turkish Wikipedia Corpora**. This dataset consists of three subsets of Wikipedia for English, German, and Turkish Wikipedia and consisting of 116M, 162M, and 52M tokens. These corpora are used for training unsupervised word embeddings and for training a language model.

- **English, German, and Turkish Word Similarity Pairs**. This dataset consists of collections of annotated word-similarity pairs in three languages. For English, we evaluate on the WS-353 data, a collection of 353 pairs of English words that have been assigned similarity ratings by human annotators, SimLex, a collection of 999 word pairs annotated via Amazon Mechanical Turk, and finally the Stanford Rare Words similarity set (RW) consisting of 2034 rare word pairs. [43, 44, 45]. For German, we

| Dataset | English | | | German | | | Turkish | | |
|---|---|---|---|---|---|---|---|---|---|
| **Method** | P | R | F1 | P | R | F1 | P | R | F1 |
| BPE | 0.5527 | 0.3989 | 0.4634 | 0.5637 | 0.4131 | 0.4768 | 0.7626 | 0.2808 | 0.4104 |
| ULM | 0.7473 | 0.5992 | 0.6651 | 0.5827 | 0.5040 | 0.5405 | **0.8731** | 0.3216 | 0.4701 |
| Morfessor | 0.7537 | 0.6513 | 0.6987 | **0.6803** | 0.5616 | 0.6153 | 0.69104 | 0.3710 | 0.4828 |
| MorphMine-NF | 0.8255 | 0.6503 | 0.7275 | 0.5717 | **0.7520** | 0.6399 | 0.5894 | 0.5024 | **0.5424** |
| MorphMine | **0.8345** | **0.6977** | **0.7600** | 0.6014 | 0.7373 | **0.6624** | 0.5341 | **0.5497** | 0.5417 |

**Table 2.4:** Morpheme Segmentation Performance.

operate on canonical translations of the the WS-353 and SimLex datasets [46]. For Turkish we evaluate on the AnlamVer word similarity dataset consisting of 500 word-pairs annotated by 12 human annotators [47].

- **English, German, and Turkish Word Analogies**. Collections of annotated word analogies in three languages. For English, we evaluate on the Google analogy dataset consisting of 19544 analogy question pairs where $8,869$ are semantic and $10,675$ syntactic (i.e. morphological) questions. [20]. For German, we operate on the German translation of the English Google analogy dataset [48]. For Turkish, counterparts of the Google analogy question set was created and contains over 2K analogy tasks.

The vocabulary lists and gold-standard segmentations are used to evaluate each method's ability to extract human-verified morphemes in an unsupervised manner. The human-curated word analogies and word similarity pairs help verify the effect of incorporating various morphemes in the unsupervised word embedding process. Finally, the Wikipedia corpora subsets are used to train the morpheme-enriched word embeddings and evaluate the benefit of morpheme enrichment on a downstream language modeling task.

**Baselines**

As a baseline for segmentation, we utilize a unigram language model segmentation of "word-pieces" and byte-pair encoding segmentation as described in the related work [31, 30]. We also compare against a state-of-the-art unsupervised morpheme segmentation tool Morfessor [9]. Finally, we compare against a variant of MorphMine that forgoes global consistency whereby each word is re-segmented after recomputing morpheme counts after the initial segmentation.

For baseline embedding methods, we utilize FastText, a proposed variation of the Skip-Gram objective that utilize subword-level information, and modify FastText to incorporate each method's segmentations to enrich word embedding. We enrich FastText with each of the morpheme segmentation baselines to compare against MorphMine enriched emebeddings. With no morpheme enrichment, FastText formulation means that it reduces to Word2Vec which we also compare against.

| Method | English | | | | German | | Turkish |
|---|---|---|---|---|---|---|---|
| **Dataset** | **WS-353** | **SimLex** | **RW** | **RW-OOV** | **WS-353** | **SimLex** | **AnlamVer** |
| SkipGram | 0.72 | **0.28** | 0.36 | – | 0.58 | 0.26 | 0.45 |
| BPE | 0.72 | **0.28** | 0.41 | 0.33 | 0.59 | **0.28** | 0.47 |
| ULM | **0.74** | **0.28** | 0.41 | 0.35 | 0.60 | **0.28** | 0.47 |
| FastText | 0.70 | 0.26 | 0.34 | 0.32 | 0.58 | 0.26 | 0.46 |
| Morfessor | **0.74** | **0.28** | 0.44 | 0.35 | 0.60 | **0.28** | 0.48 |
| MorphMine | **0.74** | **0.28** | **0.46** | **0.42** | **0.62** | **0.28** | **0.49** |

**Table 2.5:** Multilingual word similarity.

### 2.5.1 Subword Extraction Accuracy

We evaluate each morpheme segmentation algorithm at identifying human-annotated segmentations in three languages: English, Turkish, and German. We report precision, recall and F1 scores for each method. When evaluating, true-positives are indicated with a valid exact match between the extracted morpheme and the gold-standard.

In Table 2.4, we report the performance of each segmentor at successfully extracting human-annotated morphemes. As both BytePair Encoding and Unigram-LM require a morpheme vocabulary size parameter, for these methods, we perform a parameter sweep and report results from the highest performing run. Across all three languages, variants of MorphMine outperform with respect to F1 score. Further analysis shows this is primarily due to a higher recall. In comparison to MorphMine without global refinement, we see that implementing global refinement generally improves performance as seen in English and German and in the case of Turkish, performance between the MorphMine variants were overall comparable.

### 2.5.2 Word Similarity Task

We evaluate the embeddings on a word similarity task. The ground truth data consists of pairs of words and a human-annotated similarity score averaged across all human evaluations. The scores are computed via the cosine similarity between each word's vector representation and results are quantified through Spearman's rank correlation coefficient between the gold standard and the cosine similarity score. To evaluate performance of the morpheme-based embeddings to infer OOV words, we evaluate similarity on an English rare-words similarity dataset.

As seen in Table 2.5, subword-based methods that utilize morpheme and subword level

information outperform SkipGram that forgoes any. Additionally, methods that discriminately generate these morphemes outperform FastText that indiscriminately generate all subwords. Finally, while most subword enriched embeddings perform well on word similarity, MorphMine shines particularly in the similarity task on rare words where it outperforms all baselines. This is likely because MorphMine generates morphemes at multiple granularity which is more likely semantically link a rare word to a frequent word via a semantically-meaningful morpheme. This performance gap is even higher for out-of-vocabulary words were MorphMine significantly outperforms all other baselines at the word similarity task.

### 2.5.3 Word Analogy Task

We next evaluate on a word analogy task of the form "$A$ is to $B$" as "$C$ is to $D$", where $D$ is predicted from the vocabulary based on its embedding vector. We use analogy datasets used in previous literature for English, German, and Turkish embedding evaluation [20, 48, 49].

| **Dataset** | **English** | | **German** | | **Turkish** |
|---|---|---|---|---|---|
| **Method** | **Sem** | **Syn** | **Sem** | **Syn** | **Sem+Syn** |
| SkipGram | **68** | 65 | **63** | 46 | 41 |
| BPE | 65 | 68 | 61 | 50 | 43 |
| ULM | 67 | 70 | 62 | 51 | 43 |
| FastText | 52 | 75 | 59 | **53** | 43 |
| Morfessor | 64 | 75 | 61 | 52 | 44 |
| MorphMine | 67 | **78** | 61 | **53** | **47** |

**Table 2.6:** Word analogies.

As seen in Table 2.6, embeddings that utilize subword information perform better at syntactic analogies than SkipGram word embeddings without subword information. This does not extend to semantic analogies whereby utilizing subword-information seems to cause a deterioration in performance. This is intuitive as words without valid morphemes learn noisy embeddings when false morphemes are identified and used to enrich their representation. This is seen in the performance gap between FastText and SkipGram on semantic analogies whereby FastText's large number indiscriminate subwords degrades the quality of the final embedding. This degradation is mitigated by utilizing more-refined morpheme methods such as BytePair Encoding, Unigram-LM, Morfessor, and MorphMine. Overall, embeddings enriched with MorphMine morphemes demonstrate superior syntactic performance to all baselines while demonstrating comparable semantic performance to SkipGram.

| Dataset | English | German | Turkish |
|---|---|---|---|
| **Method** | **Perplexity** | **Perplexity** | **Perplexity** |
| SkipGram | 159 | 381 | 996 |
| BPE | 157 | 375 | 955 |
| ULM | 157 | 372 | 952 |
| FastText | 158 | 376 | 972 |
| Morfessor | 155 | 370 | 948 |
| MorphMine | **154** | **367** | **940** |

**Table 2.7:** Language modeling task

This supports our intuition that utilizing more subwords is useful, but only when they are of high quality; indiscriminately generating all enumerations of subword degrades quality.

### 2.5.4 Language Modeling Perplexity

As recent embedding evaluations have stressed the importance of evaluating embeddings not only on artificial tasks such as word similarities but also on downstream tasks, we evaluate on a downstream language modeling task [50]. We generate a language model with embedding vectors from the Wikipedia corpora and then evaluate by computing the perplexity on a held-out portion of the corpus *unseen in both the embedding phase and modeling phase*. We use an LSTM with two hidden layers, 600 hidden units per layer regularized with dropout with 0.2 probability, unrolled for 35 steps, and 20 batch size. Parameters are learned using Adagrad with a gradient clipping of 1 for 10 epochs. Each instance is trained on 80% of the data with a 10% test and 10% validation set.

The results are summarized in Table 2.7. Because experiments have minimal data cleaning do not drop infrequent or OOV words, the resulting perplexity is relatively higher than cleaned-datasets but directly comparable among the differing methods [36]. We observe that across all segmentation-based morpheme-enriched embeddings perform better in language modeling over traditional skip-gram. In contrast, FastText's indiscriminate enumeration of all possible morphemes appears to perform much poorer in this task. Finally, MorphMine outperforms the other morpheme enriched baselines. This may be due to MorphMine utilizing morphemes of mutliple granularity which closely capture semantic meaning of rare and OOV words at the largest granularity.

### 2.5.5 Segmentation Case Study

In Table 2.8, we present hand-selected segmentations. Unlike other methods, MorphMine identifies large morphemes shared among words in the vocabulary in addition to the more

| Word | BPE | ULM | Morfessor | MorphMine |
|---|---|---|---|---|
| vandalism | van + dal + ism | van + dal + ism | van + dal + ism | vandal + ism |
| truncate | trun + cate | trun + cate | truncate | truncat + e |
| truncated | trun + cat + ed | trun + cat + ed | truncat + ed | truncat + ed |
| truncating | trun + cat + ing | trun + cat + ing | truncat + ing | truncat + ing |
| troubleshooting | trouble + shoot + ing | trouble + shoot + ing | trouble + shoot + ing | troubleshoot + ing + trouble + shoot |

**Table 2.8:** Select segmentations from different subword segmentation algorithms.

frequent smaller morphemes. For example, the words *"truncate", "truncated", "truncating"* all share a common root, but all methods except for MorphMine are reluctant to identify *"truncat"* as a valid morpheme by removing 'e' from *truncate*. As such, all other methods fail semantically link these three words. Additionally, for '*vandalism*', most methods attempt to recognize "*van*" as a morpheme as it is a valid word, while MorphMine's parsimony criterion merges this into "*vandal*", which although not present in the vocabulary, is a valid word. Finally, given words such as *"troubleshooting"*, MorphMine's segmentation at multiple granularities captures "*troubleshoot*", which all other methods further decompose, losing much semantic meaning.

### 2.5.6 Scalability

From a high-level perspective, MorphMine consists of two separate steps: (1) mining and learning a high-quality candidate morpheme set from an input vocabulary and (2) utilizing the learned model to segment each word into morphemes. We can empirically estimate the expected runtime of each step of MorphMine by analyzing runtime as a function of input size. To this end, we select increasing subsets of the input dataset and compute the runtime of running MorphMine on each subset. To accurately measure the efficiency of each component of MorphMine, we measure the runtime of the morpheme candidate mining and morpheme segmentation independently.

As seen in Figure 2.4, mining the morpheme vocabulary appears to grow linearly with vocabulary size. We verify this by computing the coefficient of determination, $R^2$ to show how well a linear function fits the data. Morpheme mining and segmentation regressions yielded an $R^2$ of 0.989 and 0.991 respectively. This strongly suggests a linear relationship between input vocabulary size and runtime. As empirically Heap-Herdan's law has shown that vocabulary grows sublinearly in relation to corpus size, these results indicate that performing MorphMine segmentation on an input vocabulary as a preprocessing step adds negligible computational overhead [51].

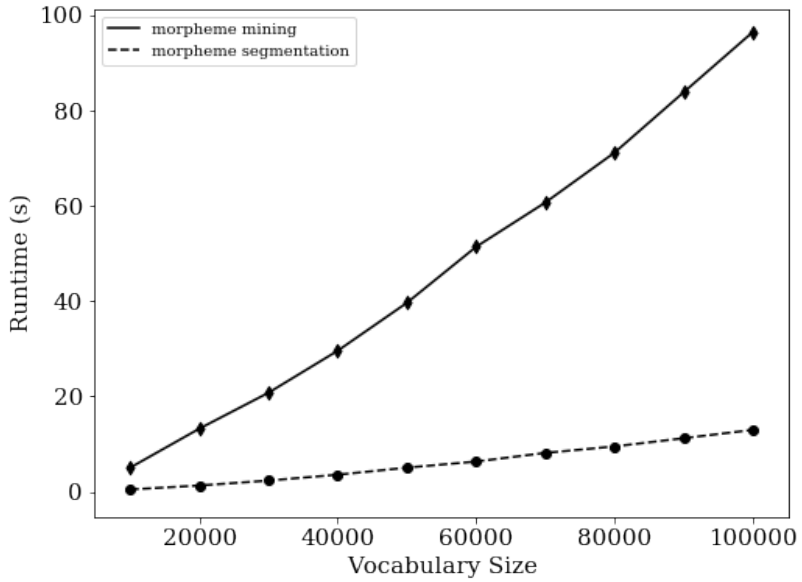



**Figure 2.4:** Decomposition of morpheme segmentation algorithm into unsupervised morpheme mining then vocabulary segmentation.

## 2.6 DISCUSSION

In this study, we propose a pattern-mining method of segmenting vocabulary into smaller morphemes and demonstrate experimentally on three languages that the method recovers ground-truth morphemes beyond state-of-the-art. By integrating the morphemes in a popular subword-enriched embedding algorithm, we verify that semantically-meaningful morphemes at multiple granularity can benefit word embeddings as evidenced through superior performance on a word analogy and word similarity task. This is especially true for inferring embeddings for infrequent or out-of-vocabulary words. Finally, we demonstrate that enriching embeddings with high-quality morphemes improves language modeling as evidenced through better held-out perplexity on a language modeling task.

One natural extension to MorphMine is to utilize a small amount of labels, either through weak, distant, or direct supervision and use this supervision to guide segmentation. This would allow flexibility to each language or domain's idiosyncrasies. Followup work can utilize various low-resource NLP tools such as part-of-speech tagging to further enrich the resultant embedding vectors. Finally, further tasks with morphemes such as sequential modeling applied to entity recognition and typing can be evaluated on morphologically-rich domains such as scientific research and biomedical corpora.

# CHAPTER 3: SUPERVISED SUBWORD MINING FOR ENRICHING WORD REPRESENTATIONS

In this chapter, we continue investigating the subword extraction problem. Within the context of Semitic languages such as Arabic or Hebrew, we identify a form of subwords that are non-contiguously located with a word and explain why they are more difficult to extract over contiguously situated subwords. Due to their non-contiguous nature, the unsupervised morpheme extraction technique utilized in MorphMine fails to extract these key morphemes. To this end, we propose a novel supervised technique for extracting these non-contiguous subwords. Experiments demonstrate that the proposed extraction framework effectively decomposes words into ground-truth morphemes. We further measure the quality of the decomposition by utilizing the extracted morphemes to enrich word embeddings and perform downstream embedding evaluation tasks.

## 3.1 INTRODUCTION

The Semitic languages are a language family commonly spoken throughout North Africa, the Horn of Africa, the Arabian peninsula, and the regions between. With approximately 500 million speakers, the proliferation of large online text collections of such news articles, social media, digitized literature, and web blogs has created a wealth of data offering challenges and opportunities for semantic understanding of Semitic texts. In these languages, a majority of words are derived from a small number of mostly triliteral consonantal roots, with some quadriliteral roots and a trace number of biliteral and quintliteral roots. It is estimated that two of the most prominent Semitic languages, Arabic and Hebrew, possess approximately 10,000 and 3,000 roots, respectively [52, 53]. As such, root identification of a given Semitic word is often an important task in morphological analysis and the first step to morphological decomposition. Morphological analysis of Semitic languages poses a unique challenge to traditional NLP techniques due to the non-contiguous morphology inherent in these languages. This morphology is best described as the application of a *pattern* resulting in the interdigitation of morphemes within a single root to form derivative words [54]. This fusional morphology allows for many surface form words derived from the same single root, but with different, yet abstractly-related semantic meanings depending on constituent morphemes. Because many surface words can be formed through this root and pattern word formation process, and the root's characters may not necessarily be contiguously situated within each resultant surface word, morpheme boundaries are often difficult to identify.

Unlike other fusional languages, the Semitic languages are unique in that the word forma-

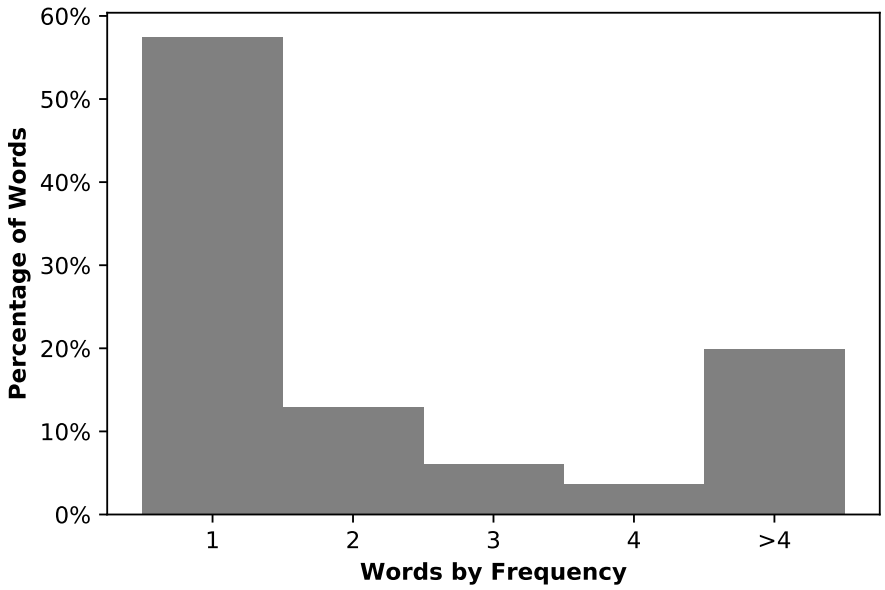



**Figure 3.1:** Word distribution in Arabic Wikipedia corpus.

tion process follows a highly-structured process of adding vowels and consonants to roots. This word formation process consists of a fixed number of slots for different morphemes, which are fixed in their position and order relative to each other. As such, these languages contain significant sequential (albeit not necessarily contiguous) substructure. In this work, we propose to leverage this sequential substructure to improve the root extraction process and morphological decomposition.

| ***Word*** | ***Translit.*** | ***Meaning*** | ***Pref.*** | ***Suff.*** | ***R-1*** | ***R-2*** |
|---|---|---|---|---|---|---|
| كتبت | KTBT | she wrote | N/A | T | N/A | N/A |
| كاتب | KĀTB | writer | N/A | N/A | Ā | N/A |
| كتاب | KTĀB | book | N/A | N/A | N/A | Ā |
| الكتاب | ALKTĀB | the book | AL | N/A | N/A | Ā |
| مكتب | MKTB | desk | M | N/A | N/A | N/A |
| مكتبة | MKTBA | library | M | A | N/A | N/A |

**Table 3.1:** Common Roots

Morphological analysis is essential in working with Semitic languages as well as other highly-inflectional languages due to data sparsity. For instance, previous research has shown that many text corpora demonstrate long-tail distributions in relation to word frequency.

This long-tail often results in corpora with many infrequent words, with $40\% - 60\%$ of words appearing just once [2]. We can verify this for Arabic in Figure 3.1, where, on a Wikipedia monolingual Arabic corpus (described in Section 3.5.1), approximately 80% of words occur fewer than five times and 60% occur once. To process such long-tailed corpora, it is necessary to exploit finer-granularity, highly-shared substructures between words that can be used to infer semantic meaning. In Table 3.1, we look at a selection of Arabic words sharing the common root –ك ت ب– (*transliteration K-T-B*), which means to *"write"*. These words are formed by appending different prefixes, suffixes, and other templatic interleavings of morphemes within the root. Despite the many surface words, the derivations share a semantic relationship based on the root, as well as other concatenative and interdigitated templatic morphemes. Additionally, as seen in the example, the root word's characters are not necessarily contiguous within the word; this is due to the non-concatenative templatic process whereby morphemes are inserted between characters of the root as part of the word formation process. Finally, not all characters in the root are necessarily found in the final surface-form of the word as some root characters can be dropped. Traditional concatenative morphological analyzers struggle to identify and extract roots precisely because root word characters are not necessarily contiguous or even present in the surface word.

To address these challenges, we present a supervised root extraction algorithm that, given a word, directly extracts the root with high accuracy. Given this root and the original word, we demonstrate how the templatic pattern-based word formation process that transforms the root to the original word can be used for further morphological decomposition. Our root extraction method differentiates itself from other methods in three ways: (1) It is fully data-driven, without any reliance on human-curated patterns; (2) it directly extracts word roots without stripping dictionary affixes, which can lead to incorrect roots when false affixes are stripped; and (3) by applying a novel sequence-to-sequence (seq2seq) model with a constrained decoding mechanism that leverages shared sequential semantics in the label (root) and input (word) space, it outperforms standard multiclass classification algorithms and achieves better generalization performance.

We demonstrate that our method outperforms unsupervised rule-based root extraction methods [55, 56, 57] and our seq2seq classifier outperforms general multiclass classifiers [58, 59]. As a testament to the utility of root extraction, we demonstrate how one can leverage the root information alongside a simple slot-based morphological decomposition to improve upon word embedding representations as evaluated through word similarity, word analogy, and language modeling tasks.

## 3.2 RELATED WORK

With the growth of the internet and the digitization of Arabic and other Semitic corpora, prior work has extensively studied root extractors with the goal of improving document retrieval [60, 61].

Early approaches to the problem of Arabic root extraction were predominantly unsupervised methods. Some researchers developed stemmers that remove some prefixes and suffixes while ignoring the templatic, interleaved morphemes within stems. A few of these methods relied on pattern matching and prefix/suffix pruning in order to extract roots [55, 56]. These methods may fail to identify the roots in many nouns and, like all prefix and suffix stripping algorithms, fail to correctly extract non-contiguous roots. Similar methods operate by removing not only prefixes and suffixes, but also "extra letters" until the triconsonantal roots remain [62]. This method, however, may incorrectly remove many letters that are part of the root. Another of these models achieves high accuracy by incorporating sentence-level context and inferred syntactic categories into a parametric Bayesian model [63]. Our model forgoes these context features as it attempts to identify the root solely on the word itself. Additionally, this method cannot model non-contiguous roots, of which Semitic languages have many. Other unsupervised methods utilize dictionaries to select the characters from within words [52, 64, 65]. Another line of research leverages the templatic nature for human-constructed rule-based constraints [66, 67, 68]. Finally, methods have been proposed that utilize both a root dictionary and rule-based templatic constraints [69].

Supervised methods have been developed for identifying Hebrew roots by combining various multiclass classification models with Hebrew-specific linguistic constraints [70]. This same technique was extended to extract both Arabic and Hebrew roots [53]. While these supervised methods effectively address the non-contiguous nature of Semitic roots, they fail to leverage the sequential structure of the root label space. We show that such methods that forgo the sequential structure in the label space underperform on words with rare roots. Additionally, these methods are only applied to triconsonantal leaving out many biconsonantal and quadriliteral roots.

Sequence-to-sequence models have been utilized for learning to map sequences to other sequences and predominantly applied to machine translation [71], with later variations of these models enhanced with attention mechanisms [72]. While LSTM variants have been dominant, previous work has shown that GRU-based models perform comparably to LSTM-based models with superior train time [59]. More recent work has investigated character-level language models in order to handle the many out-of-vocabulary (OOV) words in morphologically rich languages [73]. Such methods have shown large improvements in language

modeling across many morphologically rich languages. While such methods share the same character-level input space as does our own method, they ignore the sequential nature in the target class. Closely related to our model, constrained sequence-to-sequence models have been used for sentence simplification forcing the model to select simple words [74]. Similar approaches have been used for constrained image captioning [75]. Our model differs in that it constrains not only on specific vocabulary, but on specific sequences.

## 3.3 ROOT EXTRACTION FRAMEWORK

We introduce a framework for extracting the root from templatic words within the Semitic family. The proposed framework leverages the shared sequential semantics in both the word and root space to more accurately extract root morphemes.

### 3.3.1 Preliminaries

The input is a set of word-root pairs $W$, $R$, consisting of $|W|$ words and $|R|$ roots where $|W| = |R|$ and $W = w_1, \ldots, w_{|W|}$ and $R = r_1, \ldots, r_{|R|}$. In addition, the $j^{th}$ word $w_j$ is a sequence of $|w_j|$ characters: $c_{w_j,i}, i = 1, \ldots, |w_j|$. For convenience we index all the unique characters that compose the input vocabulary with $C$ characters and $c_{w,i} = x$, where $x \in \{1, \ldots, C\}$ means that the $i^{th}$ character in $w^{th}$ word is the $x^{th}$ character in the character vocabulary. Similarly the $k^{th}$ root, $r_k$ corresponding to the $j^{th}$ word $w_j$ is a sequence of $|r_k|$ characters: $c_{r_k,i}, i = 1, \ldots, |r_k|$. Given the input, the goal is to learn a function, $\mathcal{F} : W \rightarrow R$ that maps an input word onto its correct Semitic root.

### 3.3.2 Constrained Seq2Seq Root Extraction

Our main innovation and contribution is a unique way of extracting roots by utilizing seq2seq models for multiclass classification. While many methods traditionally approach root extraction through unsupervised application of templates or traditional supervised multiclass classification algorithms, we posit that the shared semantics between words and roots merits a different approach. As such, we apply a hybrid approach between multiclass classification and seq2seq models for root extraction. By constraining the outputs of the seq2seq models to the dictionary table of roots, the algorithm becomes a sequential multiclass classification model that implicitly leverages shared sequential substructure in both the input space and in the label space.

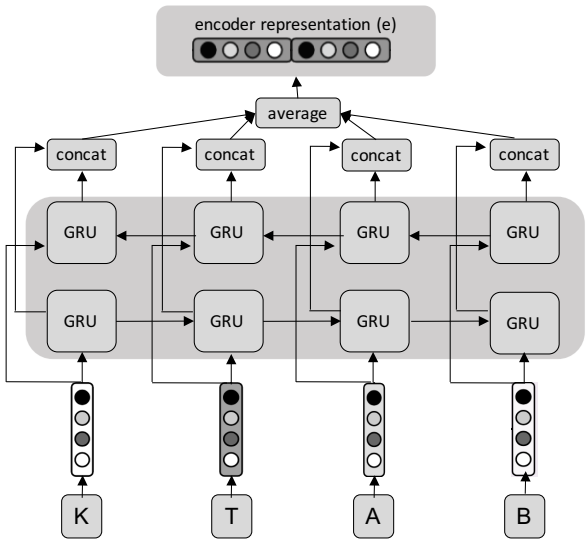



**Figure 3.2:** Encoder Network

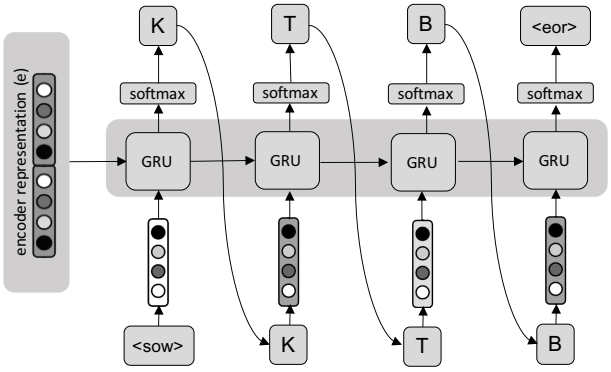



**Figure 3.3:** Decoder Network

**Figure 3.4:** Sequence-to-sequence root extraction.

Encoder Network

As seen in Figure 3.2, we begin with an encoder network that takes a word as input. Each of the input word's characters (from a total of $C$ possible characters) is associated with a vector $c \in \mathbb{R}^d$. Using word, KTĀB from Table 3.1, the input becomes vector $[c_0, c_1, c_2, c_3] \in \mathbb{R}^{d \times 4}$. We then run this sequence of embedding vectors through both directions of a bi-directional GRU (BiGRU) and concatenate the resulting hidden vectors from each pass. Finally, we average the concatenated hidden vectors of the BiGRU across all time-steps. This serves as the encoder representation of the input word, which we denote as $e$. The encoding is then fed into a decoder network that attempts to generate the most likely root for the word.

Decoder Network

In Figure 3.3, the decoder takes the encoder representation $e$ that captures the input word and predicts a root word. This is done by feeding $e$ and a special "start-of-word" character $\langle \text{sow} \rangle$ as the input. A GRU computes the next hidden state $h_0 \in \mathbb{R}^h$. A scoring function is then applied, resulting in an output the size of the character vocabulary, $C$. This function: $\mathbf{g} : \mathbb{R}^h \to \mathbb{R}^C$, is then softmaxed to obtain a valid probability distribution over characters for each hidden state. The decoding stops when the predicted root is terminated with a special "end-of-root" token $\langle \text{eor} \rangle$.

Constrained Beam Search

Traditional decoders select the best character at each step to feed into the next time step of the RNN. However, this decoding maps the input sequence into an infinite space of possible output sequences and, as such, may result in an *invalid root* that is not part of the dictionary set of roots. As such, we propose an alternative output that restricts the decoder, forcing the decoded sequence to map onto a root within the valid roots set.

We realize this constraint by modifying the decoding scheme itself. During decoding, a greedy approach is often used where the single best character output is selected and propagated to later time steps. This greedy approach may not only lead to suboptimal output sequences, but also result in invalid sequences (not corresponding to any class). This can be circumvented using a beam search decoding scheme. When decoding to obtain the predicted roots, instead of utilizing the character with the highest probability at each step, the top $k$ characters are considered at each step. As such, at each new time-step, for each of the $k$ hypotheses, there are $C$ possible choices. The top $k$ are then once again selected and this process is applied to each time step. Once all candidate roots reach their special $\langle \text{eor} \rangle$ token, the most probable root is selected. To tailor beam search to root extraction from

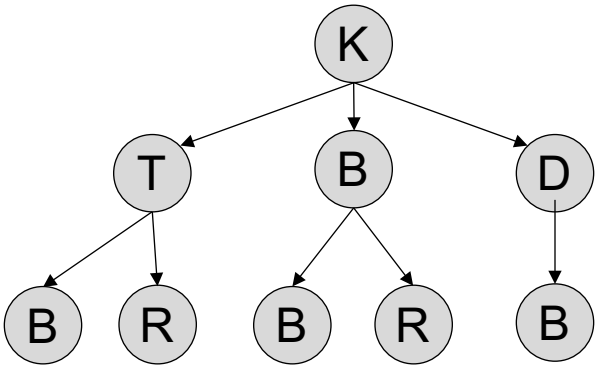



**Figure 3.5:** Constraint Trie

| Candidate Roots |
|---|
| ~~K - T - A~~ |
| K - T - B |
| ~~K - T - A - B~~ |
| K - T - R |
| ~~K - T - B - B~~ |

**Figure 3.6:** Candidate root pool.

a dictionary of roots, we seek to modify beam search by enforcing the linguistic sequential constraints present in the label root set. This leverages our classification tasks's relatively small and enumerable root label set, contrasted with an unbounded sequence as found in machine translation models. Simultaneously, by using a decoder, the model exploits the task's sequential structure by generating the target label character-by-character. We utilize the target roots as guidance for the decoding process in order to implement this sequential prediction. We demonstrate on a toy example in Figure 3.5, where by storing all the possible target roots in a trie data structure (a.k.a a prefix tree), invalid roots can be pruned during the decoding process. For example, as seen in Figure 3.6, during a typical beam-search

process, the top $k$ candidate characters are selected. By cross-referencing the current prefix of the root with the trie storing all valid roots, many invalid roots can be pruned. As such, we can enforce that the top-k selections all correspond to valid prefixes present in the target roots. This strictly improves overall extraction accuracy over traditional beam search.

## 3.4 TEMPLATIC WORD EMBEDDINGS

As the Semitic languages are templatic, there exist fixed slots that can contain morphemes. Given the correct root for a word identified as described in Section 3.3, we introduce a simple slot-based template. We indicate how to identify these slots within a word utilizing the Semitic root. Finally, we demonstrate how the morphemes within these slots, along with the root, can be utilized to enrich distributed word representations.

### 3.4.1 Morphological Decomposition

We posit that each word possesses a fixed number of slots allocated to certain morphemes, whereby the slots are fixed in their position and order relative to each other. As demonstrated in Table 3.1, in addition to the root word, we propose a simplified template that consists of four slots – two concatenative (prefixes and suffixes) and two non-concatenative (morphemes interdigitated within the stem). While we demonstrate the simplicity of identifying these within Arabic, this same template-based structure can, without loss of generality, be trivially created for other members of the Semitic family.

**Example 3.1 (Stem, Prefix, and Suffix Identification)** *For the root K-T-B, we can identify the consecutive characters that encompass the full root.*

$$AL + [KT\bar{A}B] + EEN \tag{3.1}$$

$$\text{ال} + [\text{كتاب}] + \text{ين} \tag{3.2}$$

*The characters grouped together by [] form the stem, the smallest consecutive set of characters containing the full root. Any characters not falling within the stem are, respectively, the prefixes and suffixes.*

As seen in Example 3.1, given the root, the stem can be identified as the shortest contiguous substring containing the root in correct order. Once the stem is identified, the

two concatenative slots containing prefix and suffix are trivially identified by selecting the remaining affixes after removing the stem. The non-concatenative slots can be found interdigitated within the word stem whose boundary is demarcated by the root. Given the stem (as shown in square brackets in Example 3.1) and the root, these interdigitated slots can be identified as follows:

**Example 3.2 (Interdigitated Slots)** *Given a stem containing the core root K-T-B, the candidate slots are as follows.*

*In stem, $K\bar{A}TB$, $\bar{A}$ occurs in the first slot.*
*In stem, $KT\bar{A}B$, $\bar{A}$ occurs in the second slot.*

*If a contiguous morpheme occurs after the first character in the root by before middle characters, it is a slot-1 addition. If after the middle character(s) of the root, it is slot-2.*

Example 3.2 shows the identification of interdigitated slots within the stem. Once again, it is evident that correct extraction of the root is essential to correct identification of the slot positions within the word. In the next subsection we demonstrate how these extractions can be systematically leveraged to enrich distributed word representations in these templatic languages.

### 3.4.2 Morpheme-Enriched Embeddings

To demonstrate the utility of templatic subword extractions, we demonstrate how enriching word embeddings with these morphemes can improve word representations by providing parameter-sharing between words sharing common substructure. With this motivation, we propose TemplaticVec, an intuitive extension to FastText [36], that utilizes the templatic decomposition of semantically-meaningful roots, affixes, and interdigitated morphemes for representation enrichment. By using these structures as embedding base units by and combining them to construct a word's distributed vector representation, the resultant word embeddings are robust to infrequent word-induced data-sparsity and can be constructed on many out-of-vocabulary (OOV) words. We begin with a brief review of FastText, and then demonstrate how one can naturally integrate roots as well as concatenative and templatic morphemes in place of FastText's standard naive subwords. FastText utilizes the skip-gram objective with negative sampling yielding the following objective (for simplicity,

$\ell(x) = \log(1 + \exp(-x)))$:

$$\sum_{x=1}^{|W|} \Big[ \sum_{c \in \mathcal{C}_x} \ell(s(w_x, w_c)) + \sum_{t \in \mathcal{N}_{x,c}} \ell(-s(w_x, t)) \Big] \tag{3.3}$$

In the above equation, $w_x$ is the $x^{th}$ word in the corpus, $\mathcal{C}_x$ denotes the set of context words within a predefined window of word $w_x$, and $\mathcal{N}_{x,c}$ denotes the set of negative examples sampled from outside the context window.

The scoring function is then adapted to incorporate subword information as follows:

$$s(w_x, w_c) = \sum_{m \in w_x} \mathbf{z}_m^\intercal \mathbf{v}_c \tag{3.4}$$

In the above equation, each $\mathbf{z}_m$ denotes a subword embedding vector, so that the scoring function equates to the inner product of the summation each over subword embedding vector with the context word vector. While FastText incorporates all contiguous substrings of lengths three to seven as morphemes in the scoring function, because Semitic roots are not necessarily contiguous, two words sharing the same root may not share the same subwords using FastText. Because this important semantic morpheme is not shared among words, we posit that FastText's indiscriminate enumeration of contiguous subwords does not capture the essential semantic substructure. We claim that directly incorporating the root embedding and each slot's morpheme embeddings that have been extracted for each word and summing over these embeddings results in higher quality distributed representations. As such, similar to the approach in [22], we modify the scoring function to incorporate the extracted root and slot-based templatic information:

$$s(w_x, w_c) = (\mathbf{z}_r + \mathbf{z}_p + \mathbf{z}_s + \mathbf{z}_{r1} + \mathbf{z}_{r2})^\intercal \mathbf{v}_c \tag{3.5}$$

This modification yields a scoring function that is the inner product of the summation over the root word embedding ($\mathbf{z}_r$), prefix embedding ($\mathbf{z}_p$), suffix embedding ($\mathbf{z}_s$), as well as the two possible in-root interdigitated morphemes ($\mathbf{z}_{r1}$ and $\mathbf{z}_{r2}$).

## 3.5 EXPERIMENTS

We introduce the datasets and methods for comparison used. We then describe evaluations for root extraction and embedding quality.

### 3.5.1 Datasets and comparison methods

We use the following datasets and ground-truth labels for evaluation purposes:

- **Arabic Word & Root Pairs**: 140K words along associated with 11K roots from dictionary [76].
- **Hebrew Word & Root Pairs**. 11.5K words associated with approximately 500 roots from Wiktionary[1] and human curation.
- **Arabic Wikipedia Corpora**. Wikipedia corpus with 274K articles and 62.5M tokens and 1.26M unique words.

For baseline methods to compare against our proposed constrained seq2seq (Constrain-S2S), we evaluate against three standard multiclass classification models: (1) a standard convolutional neural network, CNN-Class, [58], a GRU model, GRU-Class, and a bi-directional GRU model, BiGRU-Class. In addition, we compare against two unconstrained seq2seq models, encoder-decoder models using GRUs, GRU-S2S and bi-directional GRUs, BiGRU-S2S. Finally, for Arabic, we evaluate against three unsupervised Arabic root-extraction algorithms from the literature: Tashaphyne, ISRI, and Khoja. To evaluate on the quality of the resultant morphological decomposition, we compare against three variants of embeddings: (1) SkipGram (2) FastText (3) RootVec (Embedding enriched with solely the root) .

### 3.5.2 Root Extraction Accuracy

To evaluate the effectiveness of our proposed seq2seq extraction of roots, we perform five-fold cross-validation evaluation of our method compared to a variety of supervised and rule-based root-extraction methods. During each cross-validation, each supervised method is trained on four-fifth of the dictionary mappings of *word to root* pairs, and evaluated on a held-out 20%.

#### General Root Extraction

We first compare the performance of each supervised extraction method on extracting roots irrespective of root frequency. In Table 3.2, we report the performance of each extractor at successfully identifying the ground-truth root in each held-out word in a five-fold cross-validation evaluation. It is apparent that the unsupervised methods under-perform at extracting the ground-truth root as compared to the supervised methods. This is likely

[1] wiktionary.org

| Method | Arabic | | Hebrew | |
|---|---|---|---|---|
| | **Accuracy** | **Std. Error** | **Accuracy** | **Std. Error** |
| CNN-Classification | .6753 | ±.0009 | .9622 | ±.0019 |
| GRU-Classification | .7539 | ±.0023 | .9591 | ±.0033 |
| BiGRU-Classification | .7548 | ±.0015 | .9629 | ±.0009 |
| GRU-Seq2Seq | .7596 | ±.0017 | .9692 | ±.0013 |
| BiGRU-Seq2Seq | .7854 | ±.0010 | .9788 | ±.0016 |
| Constrained-Seq2Seq | **.8324** | **±.0011** | **.9879** | **±.0008** |
| Tashaphyne | .2778 | 0 | - | - |
| ISRI | .4508 | 0 | - | - |
| Khoja | .4434 | 0 | - | - |

**Table 3.2:** Root Extraction Accuracy.

due to errors from human-curated patterns which possess many exceptions as well as many Semitic roots being non-contiguously situated with the word due to interdigitated morphemes. Additionally, both the CNN-based and four RNN-based multiclass classification methods severely under-perform compared to our proposed constrained seq2seq model. This verifies our intuition that leveraging the shared semantic space between the words and the target roots is essential in extraction.

Rare Root Extraction

We claimed earlier that by decomposing root classification into seq2seq classification, sequential patterns within the roots can be leveraged for root extraction. This can be useful for identifying the correct root, even when the root is infrequent or even absent from the training data. To support this claim, we report the performance of each supervised extractor at successfully identifying the ground-truth of infrequent roots (appear three or fewer times in training) and a zero-shot case where the root is not present in the training data. As our Hebrew dataset consists of frequent roots, and performance is near perfect, we report results for the Arabic dataset.

As seen in Table 3.3, the seq2seq methods greatly outperform all multiclass methods with Constrain-S2S outperforming all methods on the infrequent roots. This effect is amplified in the zero-shot case, with only the seq2seq models handling unseen roots. This demonstrates the utility in jointly learning the sequential structure in semantically-shared label (root) and word space.

| Method | Infreq. | | Zero-Shot | |
|---|---|---|---|---|
| | **Accuracy.** | **Std. Error** | **Accuracy** | **Std. Error** |
| CNN-Classification | .4823 | ±.0096 | - | - |
| GRU-Classification | .5697 | ±.0103 | - | - |
| BiGRU-Classification | .5706 | ±.0091 | - | - |
| GRU-Seq2Seq | .6074 | ±.0166 | .5389 | ±.0188 |
| BiGRU-Seq2Seq | .6231 | ±.0191 | .5532 | ±.0141 |
| Constrain-Seq2Seq | **.6929** | **±.0164** | **.6292** | **±.0160** |

**Table 3.3:** Arabic Rare Root Extraction Accuracy

### 3.5.3 Word Analogy Evaluation

Given our comprehensive dataset of Arabic roots and human-curated evaluation set of Arabic word embeddings, we show the effectiveness of enriching Arabic word embeddings with their morphological decompositions via a word analogy task. The goal of said task is to identify the best value for $D$ in analogies of the form "$A$ is to $B$ as $C$ is to $D$". After training each embedding model on the Arabic Wikipedia dataset, we use an analogy dataset [77] curated for methodological evaluation of Arabic word embeddings. We further differentiate the analogies into two categories: (1) morphemic analogies (e.g. plurals, tense or gender) where a derivational or inflectional morpheme is inserted, removed, or replaced while the root remains unchanged, and (2) semantic analogies where the root itself changes between the analogous pairs (e.g. bird is to fly as fish is to swim).

| **Embedding Model** | **Semantic** | **Morphemic** |
|---|---|---|
| SkipGram | **19.1** | 11.4 |
| FastText | 13.8 | 16.8 |
| ISRI-RootVec | 15.4 | 11.2 |
| BiGRU-Classification-RootVec | 14.2 | 11.9 |
| Seq2Seq-RootVec | 18.0 | 11.9 |
| Constrained-Seq2Seq-RootVec | 18.9 | 12.2 |
| ISRI-TemplaticVec | 15.3 | 14.5 |
| Classification-TemplaticVec | 16.3 | 16.9 |
| Seq2Seq-TemplaticVec | 17.6 | 20.2 |
| Constrained-Seq2Seq-TemplaticVec | 18.8 | **22.9** |

**Table 3.4:** Word Analogies

As seen in Table 3.4, embeddings that utilize morphemes or subword-level features perform significantly better at morphemic analogies than do SkipGram word embeddings. This does not extend to semantic analogies where all methods appear to degrade with the use of

morpheme and subword-level enrichment. This is not surprising since, under the vector algebra that is used to compute the word analogies, the summation of the morphemes used to enrich the embeddings captures morphemic relationships but not necessarily semantic ones. This can be seen in the performance gap between the morpheme-enriched embeddings and SkipGram. Unlike the other methods, Templatic embeddings based on constrained roots maintains comparable performance to SkipGram on the semantic analogies while demonstrating superior performance on the morphemic analogies.

### 3.5.4 Word Similarity

The next embedding evaluation we consider is a word similarity task. The ground truth data consists of pairs of words and a human-annotated similarity score averaged across all human evaluations from a translation of the WS-353 dataset [78]. The scores are computed via the cosine similarity between the vector representation of each word in a pair. Their results are quantified through Spearman and Pearson rank correlation coefficients.

| **Embedding Model** | **Pearson** | **Spearman** |
|---|---|---|
| SkipGram | 0.496 | 0.520 |
| FastText | 0.459 | 0.468 |
| ISRI-RootVec | 0.491 | 0.518 |
| BiGRU-Classification-RootVec | 0.492 | 0.510 |
| Seq2Seq-RootVec | 0.508 | 0.516 |
| Constrained-Seq2Seq-RootVec | 0.507 | 0.514 |
| ISRI-TemplaticVec | 0.482 | 0.501 |
| Classification-TemplaticVec | 0.474 | 0.491 |
| Seq2Seq-TemplaticVec | **0.514** | 0.529 |
| Constrained-Seq2Seq-TemplaticVec | 0.512 | **0.533** |

**Table 3.5:** Word Similarity

As seen in Table 3.5, enriching the embedding vectors with the template-based extracted morphemes substantially improves embeddings in capturing word similarity. This is in contrast with lower correlation coefficients from FastText embedding vectors, likely due to the indiscriminate generation of subwords that may degrade the overall embedding. On this task, template-based decomposition using unconstrained and constrained root extraction appears to perform similarly, yet both greatly outperform the other baselines.

### 3.5.5 Language Modeling Perplexity

Finally, we evaluate the effect of utilizing the extracted root and templatic decomposition on a downstream language modeling task. On each language model, the model quality is evaluated by computing the perplexity on a held-out portion of the corpus. The model used for language modeling is an LSTM with three hidden layers, 600 hidden units per layer, regularized with 0.2 probability drop-out, unrolled for 35 steps with a batch of 20. Parameters are learned using Adagrad with a gradient clipping of 1. We evaluate on two subsets of the Wikipedia dataset: (1) LM-1, a small subset (2) LM-2, a larger subset. LM-1 consists of 3.3M tokens and a vocabulary of 260K words while LM-2 consists of 7.6M tokens and a vocabulary of 400K unique words. Each language model instance is trained for 5 epochs on the training data. Evaluation of perplexity was computed for each model on the independent test set consisting of 900K tokens where 62K tokens were OOV in LM-1 and 27K in LM-2. Evaluation is performed after selecting the best performing iteration of the model on a validation set. While the morpheme-enriched method can generate embedding vectors for many OOV tokens, for SkipGram and instances when they cannot, an unknown token with fixed embedding is used.

| **Embedding Model** | **Perplexity** | |
|---|---|---|
| | LM-One | LM-Two |
| SkipGram | 1757 | 1075 |
| FastText | 1720 | 1069 |
| ISRI-RootVec | 1729 | 1072 |
| BiGRU-Classification-RootVec | 1731 | 1071 |
| Seq2Seq-RootVec | 1728 | 1071 |
| Constrained-Seq2Seq-RootVec | 1726 | 1071 |
| ISRI-TemplaticVec | 1728 | 1071 |
| Classification-TemplaticVec | 1724 | 1070 |
| Seq2Seq-TemplaticVec | 1718 | **1065** |
| Constrained-Seq2Seq-TemplaticVec | **1716** | **1065** |

**Table 3.6:** Language Modeling

The results are summarized in Table 3.6. Although perplexity is high, this is common for morphologically-rich languages such as Arabic as shown in [73]. It appears our constrained model's extracted roots yield a benefit over other baseline roots, yet utilizing the full decomposition outperforms all other methods, yielding lower held-out perplexity. The results also verify the intuition that morphemic decomposition is necessary to handle data-sparsity and OOV words when little training data is present, whereby perplexity is greatly reduced through the use of morpheme-based embeddings.

## 3.6 DISCUSSION

In this chapter we investigated methods for extracting Semitic roots with supervision. We demonstrate that an novel classification method that utilizes a constrained sequence-to-sequence model to directly generate the root morpheme greatly outperforms competing baselines. We further demonstrate that utilizing these decomposed morphemes *in conjunction with* the original word, better enriches word embeddings in downstream tasks such as analogies, word similarity, and language modeling. This once again confirms that utilizing multiple semantic granularity benefit downstream text mining tasks.

# CHAPTER 4: PHRASE MINING AND PHRASE-BASED TOPIC MODELING

In this chapter, we investigate the benefits of utilizing unigrams and n-grams over solely unigrams in the statistical modeling and visualization of large text corpora. More specifically, we introduce a novel unsupervised phrasal segmentation algorithm that utilizes frequent contiguous pattern mining and statistical significance tests to scalably segment large corpora into single and multi-word phrases. The proposed segmentation algorithm is purely data-drive and makes no syntactic assumptions. We then investigate integrating the segmented output from our algorithm to improve a common text-mining task: statistical topic modeling.

To fully leverage our segmented corpus we modify Latent Dirichlet Allocation to incorporate single and multi-word phrases. Our proposed phrase-based topic model utilizes both word-level and phrase level restrictions for better modeling yielding more interpretable learned topics.

## 4.1 INTRODUCTION

In recent years, topic modeling has become a popular method for discovering the abstract 'topics' that underly a collection of documents. A topic is typically modeled as a multinomial distribution over terms, and frequent terms related by a common theme are expected to have a large probability in a topic multinomial. When latent topic multinomials are inferred, it is of interest to visualize these topics in order to facilitate human interpretation and exploration of the large amounts of unorganized text often found within text corpora. In addition, visualization provides a qualitative method of validating the inferred topic model [79]. A list of most probable unigrams is often used to describe individual topics, yet these unigrams often provide a hard-to-interpret or ambiguous representation of the topic. Augmenting unigrams with a list of probable phrases provides a more intuitively understandable and accurate description of a topic. This can be seen in the term/phrase visualization of an information retrieval topic in Table 4.1.

While topic models have clear application in facilitating understanding, organization, and exploration in large text collections such as those found in full-text databases, difficulty in interpretation and scalability issues have hindered adoption. Several attempts have been made to address the prevalent deficiency in visualizing topics using unigrams. These methods generally attempt to infer phrases and topics simultaneously by creating complex generative mechanism. The resultant models can directly output phrases and their latent topic assignment. Two such methods are Topical N-Gram and PD-LDA [80, 81]. While it is appealing

| *Terms* | *Phrases* |
|---|---|
| search | information retrieval |
| web | social networks |
| retrieval | web search |
| information | search engine |
| based | support vector machine |
| model | information extraction |
| document | web page |
| query | question answering |
| text | text classification |
| social | collaborative filtering |
| user | topic model |

**Table 4.1:** Visualization of the topic of Information Retrieval, automatically constructed by ToPMine from titles of computer science papers published in DBLP (20Conf dataset).

to incorporate the phrase-finding element in the topical clustering process, these methods often suffer from high-complexity, and overall demonstrate poor scalability outside small datasets.

Some other methods apply a post-processing step to unigram-based topic models [82, 83]. These methods assume that all words in a phrase will be assigned to a common topic, which, however, is not guaranteed by the topic model.

We propose a new methodology ToPMine that demonstrates both scalability compared to other topical phrase methods and interpretability. Because language exhibits the principle of non-compositionality, where a phrase's meaning is not derivable from its constituent words, under the 'bag-of-words' assumption, phrases are decomposed, and a phrase's meaning may be lost [28]. Our insight is that phrases need to be systematically assigned to topics. This insight motivates our partitioning of a document into phrases, then using these phrases as constraints to ensure all words are systematically placed in the same topic.

We perform topic modeling on phrases by *first mining phrases, segmenting each document into single and multi-word phrases, and then using the constraints from segmentation in our topic modeling*. First, to address the scalability issue, we develop an efficient phrase mining technique to extract frequent significant phrases and segment the text simultaneously. It uses frequent phrase mining and a statistical significance measure to segment the text while simultaneously filtering out false candidates phrases. Second, to ensure a systematic method of assigning latent topics to phrases, we propose a simple but effective topic model. By restricting all constituent terms within a phrase to share the same latent topic, we can assign a phrase the topic of its constituent words.

**Example 4.1** *By frequent phrase mining and context-specific statistical significance ranking, the following titles can be segmented as follows:*

***Title 1.*** *[Mining frequent patterns] without candidate generation: a [frequent pattern] tree approach.*

***Title 2.*** *[Frequent pattern mining] : current status and future directions.*

*The tokens grouped together by [] are constrained to share the same topic assignment.*

Our TopMine method has the following advantages.

- Our phrase mining algorithm efficiently extracts candidate phrases and the necessary aggregate statistics needed to prune these candidate phrases. Requiring no domain knowledge or specific linguistic rulesets, our method is purely data-driven.
- Our method allows for an efficient and accurate filtering of false-candidate phrases. In title 1 of Example 4.1, after merging 'frequent' and 'pattern', we only need to test whether 'frequent pattern tree' is a *significant* phrase in order to determine whether to keep 'frequent pattern' as a phrase in this title.
- Segmentation induces a 'bag-of-phrases' representation for documents. We incorporate this as a constraint into our topic model eliminating the need for additional latent variables to find the phrases. The model complexity is reduced and the conformity of topic assignments within each phrase is maintained.

## 4.2 RELATED WORK

Recently many attempts have been made to relax the 'bag-of-words' assumption of LDA. These topical phrase extraction techniques fall into two main categories, those that infer phrases and topics simultaneously by creating complex generative models and those that apply topical phrase discovery as a post-process to LDA.

Methods have experimented with incorporating a bigram language model into LDA [84]. This method uses a hierarchical dirichlet to share the topic across each word within a bigram. TNG [80] is a state-of-the-art approach to n-gram topic modeling that uses additional latent variables and word-specific multinomials to model bi-grams. These bigrams can be combined to form n-gram phrases. PD-LDA uses a hierarchal Pitman-Yor process to share the same topic among all words in a given n-gram [81]. Because PD-LDA uses a nonparametric prior to share a topic across each word in an n-gram, it can be considered a natural generalization of the LDA bigram language model to n-grams and more appropriate for comparison.

Other methods construct topical phrases as a post-processing step to LDA and other topic models. KERT constructs topical phrases by performing unconstrained frequent pattern

mining on each topic within a document then ranking the resultant phrases based on four heuristic metrics [83]. Turbo Topics uses a back-off n-gram model and permutation tests to assess the significance of a phrase [82].

Topical key phrase extraction has even been applied to the social networking service Twitter [85]. Using a Twitter-specific topic model and ranking candidate key phrases with an extension of topical page-rank on retweet behavior, the method extracts high-quality topical keywords from twitter. Because this method relies on the network topology of twitter, it doesn't extend to other text corpora.

Attempts to directly enrich the text corpora with frequent pattern mining to enhance for topic modeling have also been investigated [86]. As the objective of this method is to enrich the overall quality of the topic model and not for the creation of interpretable topical phrases, their main focus is different from ToPMine.

The concept of placing constraints into LDA has been investigated in several methods. Hidden Topic Markov Model makes the assumption that all words in a sentence have the same topic with consecutive sentences sharing the same topic with a high probability [87]. By relaxing the independence assumption on topics, this model displays a drop in perplexity while retaining computational efficiency. Sentence-LDA, is a generative model with an extra level generative hierarchy that assigns the same topic to all the words in a single sentence [88]. In both of these models, the final output produced is a general topic model with no intuitive method of extracting topical phrases.

There is a large literature on unsupervised phrase extraction methods. These approaches generally fall into one of a few techniques: language modeling, graph-based ranking, and clustering [89]. Because these methods simply output a ranked list of phrases, they are incompatible with our phrase-based topic modeling which operates on partitioned documents.

## 4.3 PROBLEM DEFINITION

The input is a corpus of $D$ documents, where $d$-th document is a sequence of $\mathcal{N}_d$ tokens: $w_{d,i}, i = 1, \ldots, \mathcal{N}_d$. Let $N = \sum_{d=1}^{D} \mathcal{N}_d$. For convenience we index all the unique words in this corpus using a vocabulary of $V$ words. And $w_{d,i} = x, x \in \{1, \ldots, V\}$ means that the $i$-th token in $d$-th document is the $x$-th word in the vocabulary. Throughout this paper we use 'word $x$' to refer to the $x$-th word in the vocabulary.

Given a corpus and a number of topics as a parameter, our goal is to infer the corpus' underlying topics and visualize these topics in a human-interpretable representation using topical phrases. Statistically, a topic $k$ is characterized by a probability distribution $\phi_k$ over words. $\phi_{k,x} = p(x|k) \in [0, 1]$ is the probability of seeing the word $x$ in topic $k$, and

$\sum_{x=1}^{V} \phi_{k,x} = 1$. For example, in a topic about the database research area, the probability of seeing "database", "system" and "query" is high, and the probability of seeing "speech", "handwriting" and "animation" is low. This characterization is advantageous in statistical modeling of text, but is weak in human interpretability. Unigrams may be ambiguous, especially across specific topics. For example, the word "model" can mean different things depending on a topic - a model could be a member of the fashion industry or perhaps be part of a phrase such as "topic model". Using of phrases helps avoid this ambiguity.

**Definition 4.1** *We formally define phrases and other necessary notation and terminology as follows:*

- *A* phrase *is a sequence of contiguous tokens:*
  $P$=$\{w_{d,i}, ...w_{d,i+n}\}$ $n > 0$
- *A* partition *over d-th document is a sequence of phrases:* $(P_{d,1}, \ldots, P_{d,G_d})$ $G_d \geq 1$ *s.t. the concatenation of the phrase instances is the original document.*

In Example 4.1, we can see the importance of word proximity in phrase recognition. As such, we place a contiguity restriction on our phrases.

To illustrate an induced partition upon a text segment, we can note how the concatenation of all single and multi-word phrases in Title 1 will yield an ordered sequence of tokens representing the original title.

### 4.3.1 Desired Properties

We outline the desired properties of a topical phrase mining algorithm as follows:

- The lists of phrases demonstrate a coherent topic.
- The phrases extracted are valid and human-interpretable.
- Each phrase is assigned a topic in a principled manner.
- The overall method is computationally efficient and of comparable complexity to LDA.
- The topic model demonstrates similar perplexity to LDA

In addition to the above requirements for the system as a whole, we specify the requirements of a topic-representative phrase. When designing our phrase mining framework, we ensure that our phrase-mining and phrase-construction algorithms naturally validate candidate phrases on three qualities that constitute human-interpretability.

1. **Frequency:** The most important quality when judging whether a phrase relays important information regarding a topic is its frequency of use within the topic. A phrase that is not frequent within a topic, is likely not important to the topic. This formulation can be interpreted as a generalization of the list of most probable unigrams visualization used for LDA to a list of the most probable phrases one will encounter in a given topic.

2. **Collocation:** In corpus linguistics, a collocation refers to the co-occurence of tokens in such frequency that is significantly higher than what is expected due to chance. A commonly-used example of a phraseological-collocation is the example of the two candidate collocations "strong tea" and "powerful tea"[90]. One would assume that the two phrases appear in similar frequency, yet in the English language, the phrase "strong tea" is considered more correct and appears in much higher frequency. Because a collocation's frequency deviates from what is expected, we consider them 'interesting' and informative. This insight motivates the necessity of analyzing our phrases probabilistically to ensure they are collocations.

3. **Completeness:** If long frequent phrases satisfy the above criteria, then their subsets also satisfy these criteria. For example in the case of "mining frequent patterns", "mining frequent" will satisfy the frequency and collocation restriction, yet is clearly a subset of a larger and more intuitive phrase. Our phrase-construction algorithm should be able to automatically determine the most appropriate size for a human-interpretable phrase.

We will introduce a framework that naturally embeds these phrase requirements.

## 4.4 TOPMINE FRAMEWORK

To extract topical phrases that satisfy our desired requirements, we propose a framework that can be divided into two main parts: phrase-mining with text segmentation and phrase-constrained topic modeling. Our process for transforming a 'bag-of-words' document to a high-quality 'bag-of-phrases' involves first mining frequent phrases, and then using these phrases to segment each document through an agglomerative phrase construction algorithm. After inducing a partition on each document, we perform topic modeling to associate the same topic to each word in a phrase and thus naturally to the phrase as a whole.

The goal of our phrase mining is to collect aggregate statistics for our phrase-construction. The statistical significance measure uses these aggregates to guide segmentation of each

document into phrases. This methodology leverages phrase context and phrase significance in the construction process ensuring all phrases are of high-quality. The resultant partition is the input for our phrase-constrained topic model.

We choose the 'bag-of-phrases' input over the traditional 'bag-of-words' because under the latter assumption, tokens in the same phrase can be assigned to different latent topics. We address this by proposing a topic model PhraseLDA, which incorporates the 'bag-of-phrases' partition from our phrase mining algorithm as constraints in the topical inference process. We have derived a collapsed Gibb's sampling method that when performing inference, ensures that tokens in the same phrase are assigned to the same topic.

## 4.5 PHRASE MINING

We present a phrase-mining algorithm that given a corpus of documents, merges the tokens within the document into human-interpretable phrases. Our method is purely data-driven allowing for great cross-domain performance and can operate on a variety of datasets. We extract high-quality phrases by obtaining counts of frequent contiguous patterns, then probabilistically reasoning about these patterns while applying context constraints to discover meaningful phrases.

The phrase mining algorithm can be broken down into two major steps. First, we mine the corpus for frequent candidate phrases and their aggregate counts. We have developed a technique that can quickly collect this information without traversing the prohibitively large search space. Second, we agglomeratively merge words in each document into quality phrases as guided by our *significance measure*. We will discuss these steps in greater detail in the next two subsections.

### 4.5.1 Frequent Phrase Mining

In Algorithm 4.1, we present our frequent phrase mining algorithm. The task of frequent phrase mining can be defined as collecting aggregate counts for all contiguous words in a corpus that satisfy a certain minimum support threshold. We draw upon two properties for efficiently mining these frequent phrases.

1. Downward closure lemma: If phrase $P$ is not frequent, then any super-phrase of $P$ is guaranteed to be **not** frequent.

2. Data-antimonotonicity: If a document contains no frequent phrases of length $n$, the document does **not** contain frequent phrases of length $> n$.

The downward closure lemma was first introduced for mining general frequent patterns using the Apriori algorithm [91]. We can exploit this property for our case of phrases by maintaining a set of active indices. These active indices are a list of positions in a document at which a contiguous pattern of length $n$ is frequent. In line 1 of Algorithm 4.1, we see the list of active indices.

In addition, we use the data-antimonotonicity property to assess if a document should be considered for further mining [92]. If the document we are considering has been deemed to contain no more phrases of a certain length, then the document is guaranteed to contain no phrases of a longer length. We can safely remove it from any further consideration. These two pruning techniques work well with the natural sparsity of phrases and provide early termination of our algorithm without searching through the prohibitively large candidate phrase space.

**Algorithm 4.1:** Frequent Phrase Mining

**Input:** Corpus with $D$ documents, min support $\epsilon$
**Output:** Frequent phrase and their frequency: {(P,C(P))}

```
1  D ← [D]
2  A_{d,1} ← {indices of all length-1 phrases ∈ d}   ∀d ∈ D
3  C ← HashCounter(counts of frequent length-1 phrases)
4  n ← 2
5  while D ≠ ∅ do
6      for d ∈ D do
7          A_{d,n} ← {i ∈ A_{d,n−1} | C[{w_{d,i}..w_{d,i+n−2}}] ≥ ε}
8          A_{d,n} ← A_{d,n} \ {max(A_{d,n})}
9          if A_{d,n} = ∅ then
10             D ← D \ {d}
11         else
12             for i ∈ A_{d,n} do
13                 if i + 1 ∈ A_{d,n} then
14                     P ← {w_{d,i}..w_{d,i+n−1}}
15                     C[P] ← C[P] + 1
16                 end
17             end
18         end
19     end
20     n ← n + 1
21 end
22 return {(P, C[P]) : s.t > C[P] ≥ ε}
```

We take an increasing-size sliding window over the corpus to generate candidate phrases and obtain aggregate counts. At iteration $k$, for each document still in consideration, fixed-length candidate phrases beginning at each active index are counted using an appropriate hash-based counter. As seen in Algorithm 4.1 line 7, candidate phrases of length $k-1$ are pruned if they do not satisfy the minimum support threshold and their starting position is removed from the active indices. We refer to this implementation of the downward closure lemma as *position-based Apriori pruning*. As seen in Algorithm 4.1 lines 9 - 11, when a document contains no more active indices, it is removed from any further consideration. This second condition in addition to pruning the search for frequent phrases provides a natural termination criterion for our algorithm.

The frequency criterion requires phrases to have sufficient occurrences. In general, we can set a minimum support that grows linearly with corpus size. The larger minimum support is, the more precision and the less recall is expected.

General frequent transaction pattern mining searches an exponential number of candidate patterns [91, 93]. When mining phrases, our contiguity requirement significantly reduces the number of candidate phrases generated. Worst case time-complexity occurs when the entire document under consideration meets the minimum support threshold. In this scenario, for a document $d$ we generate $\mathcal{O}(\mathcal{N}_d^2)$ (a quadratic number) candidate phrases. Although this quadratic time and space complexity seems prohibitive, several properties can be used to ensure better performance. First, separating each document into smaller segments by splitting on phrase-invariant punctuation (commas, periods, semi-colons, etc) allows us to consider constant-size chunks of text at a time. This effectively makes the overall complexity of our phrase mining algorithm linear, $\mathcal{O}(N)$, in relation to corpus size. The downward closure and data antimonotonicity pruning mechanisms serve to further reduce runtime.

### 4.5.2 Segmentation and Phrase Filtering

Traditional phrase extraction methods filter low quality phrases by applying a heuristic "importance" ranking that reflect confidence in candidate key phrases, then only keeping the top-ranked phrases [94]. Some methods employ external knowledge bases or NLP constraints to filter out phrases [95, 94].

Our candidate phrase filtering step differentiates itself from traditional phrase extraction methods by implicitly filtering phrases in our document segmentation step. By returning to the context and constructing our phrases from the bottom-up, we can use phrase-context and the partition constraints to determine which phrase-instance was most likely intended. Because a document can contain at most a linear number of phrases (the number of terms

in the document) and our frequent phrase mining algorithm may generate up to a quadratic number of candidate phrases, a quadratic number of bad candidate phrases can be eliminated by enforcing the partition constraint.

The key element of this step is our bottom-up merging process. At each iteration, our algorithm makes locally optimal decisions in merging single and multi-word phrases as guided by a statistical significance score. In the next subsection, we present an agglomerative phrase-construction algorithm then explain how the significance of a potential merging is evaluated and how this significance guides our agglomerative merging algorithm.

### Phrase Construction Algorithm

The main novelty in our phrase mining algorithm is the way we construct our high-quality phrases by inducing a partition upon each document. We employ a bottom-up agglomerative merging that greedily merges the best possible pair of candidate phrases at each iteration. This merging constructs phrases from single and multi-word phrases while maintaining the partition requirement. Because only phrases induced by the partition are valid phrases, we have implicitly filtered out phrases that may have passed the minimum support criterion by random chance.

**Algorithm 4.2:** Bottom-up Construction of Phrases from Ordered Tokens

**Input:** Counter $C$, thresh $\alpha$
**Output:** Partition

```
1  H ← MaxHeap()
2  Place all contiguous token pairs into H with their significance score key.
3  while H.size() > 1 do
4  |   Best ← H.getMax()
5  |   if Best.Sig ≥ α then
6  |   |   New ← Merge(Best)
7  |   |   Remove Best from H
8  |   |   Update significance for New with its left phrase instance and right phrase instance
9  |   else
10 |   |   break
11 |   end
12 end
```

In Algorithm 4.2, we present the phrase construction algorithm. The algorithm takes as input a document and the aggregate counts obtained from the frequent phrase mining algorithm.

It then iteratively merges phrase instances with the strongest association as guided by a potential merging's significance measure. The process is a bottom-up approach that upon termination induces a partition upon the original document creating a 'bag-of-phrases'.

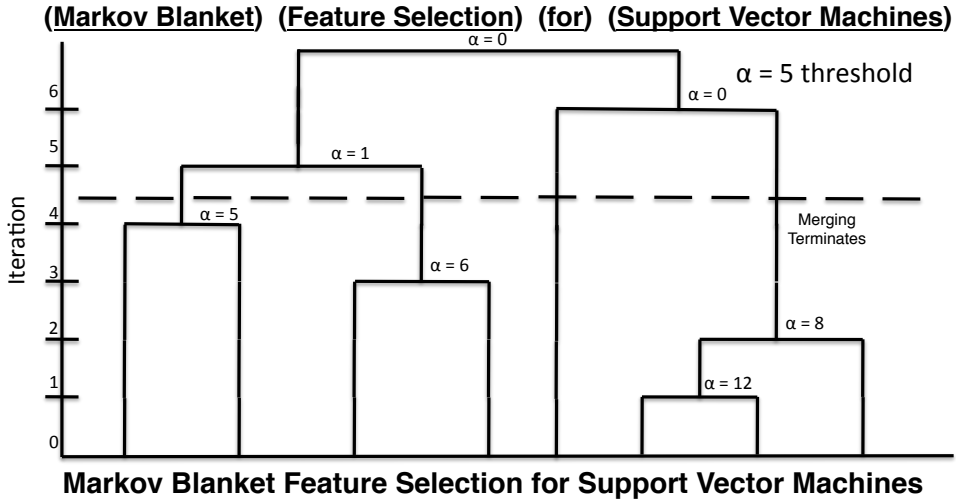



**Figure 4.1:** Bottom-up construction of a 'bag-of-phrases' on computer science title taken from DBLP.

Figure 4.1 tracks the phrase construction algorithm by visualizing the agglomerative merging of phrases at each iteration with a dendogram. Operating on a paper title obtained from our dblp titles dataset, each level of the dendogram represents a single merging. At each iteration, our algorithm selects two contiguous phrases such that their merging is of highest significance (Algorithm 4.2 line 4) and merges them (Algorithm 4.2 lines 6 - 9) . The following iteration then considers the newly merged phrase as a single unit. By considering each newly merged phrase as a single unit and assessing the significance of merging two phrases at each iteration, we successfully address the "free-rider" problem where long, unintelligible, phrases are evaluated as significant when comparing the occurrence of a phrase to the occurrence of each constituent term independently.

As all merged phrases are frequent phrases, we have fast access to the aggregate counts necessary to calculate the significance values for each potential merging. By using proper data structures, the contiguous pair with the highest significance can be selected and merged in logarithmic time, $\mathcal{O}(log(\mathcal{N}_d))$ for each document. This complexity can once again be reduced by segmenting each document into smaller chunk by splitting on phrase-invariant punctuation. Our algorithm terminates when the next merging with the highest significance does not meet a predetermined significance threshold $\alpha$ or when all the terms have been

merged into a single phrase. This is represented by the dashed line in Figure 4.1 where there are no more candidate phrases that meet the significance threshold. Upon termination, a natural "bag-of-phrases" partition remains. While the frequent phrase mining algorithm satisfies the *frequency* requirement, the phrase construction algorithm satisfies the collocation and completeness criterion.

To statistically reason about the occurrence of phrases, we consider a null hypothesis, that the corpus is generated from a series of independent Bernoulli trials. Under this hypothesis, the presence or absence of a phrase at a specific position in the corpus is a product of a Bernoulli random variable, and the expected number of occurrences of a phrase can be interpreted as a binomial random variable. Because the number of tokens $L$ in the corpus can be assumed to be fairly large, this binomial can be reasonably approximated by a normal distribution. As such, the null hypothesis distribution, $h_0$, for the random variable $f(P)$, the count of a phrase $P$ within the corpus is:

$$h_0(f(P)) = \mathcal{N}(Lp(P), Lp(P)(1 - p(P))) \tag{4.1}$$

$$\approx \mathcal{N}(Lp(P), Lp(P)) \tag{4.2}$$

where $p(P)$ is the Bernoulli trial success probability for phrase $P$. The empirical probability of a phrase in the corpus can be estimated as $p(P) = \frac{f(P)}{L}$. Consider a longer phrase that composed of two phrases $P_1$ and $P_2$. The mean of its frequency under our null hypothesis of independence of the two phrases is:

$$\mu_0(f(P_1 \oplus P_2)) = Lp(P_1)p(P_2) \tag{4.3}$$

This expectation follows from treating each phrase as a constituent, functioning as a single unit in the syntax. Due to the unknown population variance and sample-size guarantees from the minimum support, we can estimate the variance of the population using sample variance: $\sigma^2_{P_1 \oplus P_2} \approx f(P_1 \oplus P_2)$, the sample phrase occurrence count. We use a significance score to provide a quantitative measure of which two consecutive phrases form the best collocation at each merging iteration. This is measured by comparing the actual frequency with the expected occurrence under $h_0$.

$$sig(P_1, P_2) \approx \frac{f(P_1 \oplus P_2) - \mu_0(P_1, P_2)}{\sqrt{f(P_1 \oplus P_2)}} \tag{4.4}$$

Equation 4.4 computes the number of standard deviations away from the expected number of occurrences under the null model. This significance score can be calculated using the aggregate counts of the candidate phrases, which can be efficiently obtained from the frequent phrase-mining algorithm. This significance score can be considered a generalization of the t-statistic which has been used to identify dependent bigrams [96, 97]. By checking the $h_0$ of merging two contiguous sub-phrases as opposed to merging each individual term in the phrase, we effectively address the 'free-rider' problem where excessively long phrases appear significant. To address the concern that the significance score relies on the naive independence assumption, we do not perform hypothesis testing to accept or reject $h_0$. Instead we use the score as a robust collocation measure by which to guide our algorithm in selecting phrases to merge. A high significance indicates a high-belief that two phrases are highly associated and should be merged.

## 4.6 TOPIC MODELING

In the previous section, we segment a document into a collection of phrases, which provides a new representation for documents, i.e. 'bag-of-phrases'. These phrases are a group of words that appear frequently, contiguously, and occur more often than due to chance. Our insight is that *with high probability*, tokens in the same phrase should share the same latent topic.

In this section, we start with a brief review of Latent Dirichlet Allocation [98], and then propose a novel probabilistic model, PhraseLDA, which incorporates the 'constraint' idea into LDA. A collapsed Gibbs sampling algorithm is developed for PhraseLDA, and optimization of hyper-parameters is discussed. Finally, we define topical frequency for phrases, which serves as a ranking measure for our topic visualization. We list the notations used in Table 4.2, where $\mathcal{I}(statement) = 1$ if statement is true; otherwise 0. We denote $Z$ the collection of all latent variables $\{z_{d,g,j}\}$, and $W, \Theta, \Phi$ the collection of their corresponding random variables $\{w_{d,g,j}\}, \{\theta_d\}, \{\phi_k\}$.

### 4.6.1 Brief review of LDA

LDA assumes that a document is a mixture of topics, where a topic is defined to be a multinomial distribution over words in the vocabulary. The generative process is as follows:

1. Draw $\phi_k \sim Dir(\beta)$, for $k = 1, 2, ..., K$
2. For d$th$ document, where $d = 1, 2, ..., D$:

| Variable | Description |
|---|---|
| D, K, V | number of documents, topics, size of vocabulary |
| d, g, i, k, x | index for document, phrase in a doc, token in a doc, topic, word |
| $N_d$ | number of tokens in d$th$ doc |
| $G_d$ | number of phrases in d$th$ doc(after partition) |
| $W_{d,g}$ | number of tokens in g$th$ phrase of d$th$ doc |
| $\theta_d$ | multinomial distribution over topics for d$th$ doc |
| $z_{d,g,j}$ | latent topic for j$th$ token in g$th$ phrase of d$th$ doc |
| $w_{d,g,j}$ | the j$th$ token in g$th$ phrase of doc d |
| $\phi_k$ | multinomial distribution over words in topic k |
| $\mathcal{N}_k$ | $\mathcal{N}_k = \sum_{d,g,j} \mathcal{I}(z_{d,g,j} == k)$, number of tokens assigned to topic k |
| $\mathcal{N}_{d,k}$ | $\mathcal{N}_{d,k} = \sum_{g,j} \mathcal{I}(z_{d,g,j} == k)$, number of tokens assigned to topic k in doc d |
| $\mathcal{N}_{x,k}$ | $\mathcal{N}_{x,k} = \sum_{d,g,j} \mathcal{I}(z_{d,g,j} == k, w_{d,g,j} == x)$, number of tokens with value x and topic k |
| $\alpha, \beta$ | parameter of the Dirichlet distribution for $\theta_d, \phi_k$ |
| $\mathcal{C}_{d,g}$ | $\{z_{d,g,j}\}_{j=1}^{W_{d,g}}$, the collection of all latent variables in g$th$ clique(phrase) of d$th$ doc |

**Table 4.2:** Notation used in topic modeling

(a) Draw $\theta_d \sim Dir(\alpha)$

(b) For i$th$ token in d$th$ document, where $i = 1, 2, ..., N_d$:

i. Draw $z_{d,i} \sim Multi(\theta_d)$

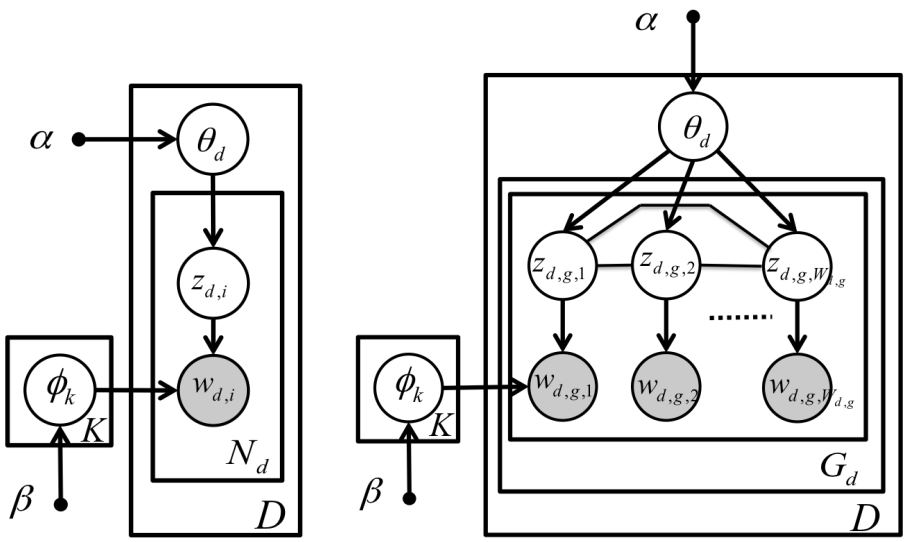
ii. Draw $w_{d,i} \sim Multi(\phi_{z_{d,i}})$

The graphical model for LDA, depicted in Figure 4.2(a), defines the joint distribution of random variables. By utilizing the conditional independence encoded in the graph, the joint distribution can be written as(we omit the hyper-parameter $\alpha, \beta$ for simplicity):

$$P_{\text{LDA}}(Z, W, \Phi, \Theta) = \prod_{d,i} p(z_{d,i}|\theta_d) p(w_{d,i}|z_{d,i}, \Phi) \prod_d p(\theta_d) \prod_k p(\Phi_k) \tag{4.5}$$

Because of the conjugacy between multinomial and Dirichlet distributions, we can easily integrate out $\{\Theta, \Phi\}$. That is,

$$P_{\text{LDA}}(Z, W) = \int P_{\text{LDA}}(Z, W, \Phi, \Theta) d\Theta d\Phi \tag{4.6}$$




(a) Bayesian network for LDA

(b) chain graph for PhraseLDA

**Figure 4.2:** In PhraseLDA, latent topic variables ($z_{d,g,j}$) in the same phrase form a clique. Each clique introduces a potential function onto the joint distribution defined by LDA. For computational efficiency, we choose a potential function which assigns same-clique tokens to the same topic

has a closed form.

### 4.6.2 PhraseLDA

LDA is built upon the 'bag-of-words' assumption, under which the order of words is completely ignored. As a result, when inferring the topic assignment $z_{d,i}$ for word $w_{d,i}$, the topic of a far-away word in the same document has the same impact as a near-by word. In section 4.5, we partition a document into a collection of phrases. We believe that the high frequency of occurrence significantly greater than due to chance, and the proximity constraint induced by contiguity are an indication that there is a stronger correlation than that expressed by LDA between the words in a phrase. This motivates our use of the chain graph to model this stronger correlation. Chain graphs are most appropriate when there are both response-explanatory relations (Bayesian networks) and symmetric association relations (Markov networks) among variables [99]. In our task, LDA models the (directed) causal relations between topics and the observed tokens, and we propose to use un-directed graph to model the stronger dependence among near-by words.

We connect the latent topic assignments in the same phrase using un-directed edges(as shown in figure 4.2). As a result, for g$th$ phrase of d$th$ document, random variables $\{z_{d,g,j}\}_{j=1}^{W_{d,g}}$

form a clique.

Every clique $\mathcal{C}_{d,g}$(or equivalently phrase) introduces a potential function $f(\mathcal{C}_{d,g})$, which should express the intuition that with high probability, $z_{d,g,j}$'s in the clique should take the same value. As a result, the chain graph defines the joint distribution over all random variables:

$$P(Z, W, \Phi, \Theta) = \frac{1}{C} P_{\text{LDA}}(Z, W, \Phi, \Theta) \prod_{d,g} f(\mathcal{C}_{d,g}) \tag{4.7}$$

where $C$ is a normalizing constant that makes the left hand side a legitimate probability distribution.

### 4.6.3 Inference

For the joint distribution defined by equation 4.7, we developed a collapsed gibbs sampling algorithm to sample latent assignment variables $Z$ from its posterior. As with LDA, the first step is to integrate out $\{\Theta, \Phi\}$:

$$\begin{aligned} P(Z, W) &= \int \frac{1}{C} P_{\text{LDA}}(Z, W, \Phi, \Theta) \prod_{d,g} f(\mathcal{C}_{d,g}) \, d\Theta d\Phi \\ &= \frac{1}{C} \left( \int P_{\text{LDA}}(Z, W, \Phi, \Theta) d\Theta d\Phi \right) \prod_{d,g} f(\mathcal{C}_{d,g}) \\ &= \frac{1}{C} P_{\text{LDA}}(Z, W) \prod_{d,g} f(\mathcal{C}_{d,g}) \end{aligned} \tag{4.8}$$

$P(Z, W)$ takes a simple closed form because $P_{\text{LDA}}(Z, W)$ does.

Ideally, the potential function in equation 4.7 expresses the strong (symmetric) influence between the words within a phrase. Suppose clique $\mathcal{C}_{d,g}$ is of size s, then $\mathcal{C}_{d,g}$ can be in any of the $K^s$ possible states, where $K$ is the number of topics. Since the normalizing constant is unknown, we need to compute a value for all $K^s$ states, and then normalize the values to get a legitimate distribution, which is computationally intractable for large $K$ and $s$. As such, we choose a specific potential function below:

$$f(\mathcal{C}_{d,g}) = \begin{cases} 1 & \text{if} \quad z_{d,g,1} = z_{d,g,2} = ... = z_{d,g,W_{d,g}} \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

This potential function coerces all variables in the clique to take on the same latent topic. Because our phrase-mining algorithm performs a constrained merging guided by a statistical significance measure, we assume that it is of high probability that the random variables in

the clique possess the same topic. As such, we adopt the potential function as specified by equation 4.9 as an approximation, which reduces the possible states of $\mathcal{C}_{d,g}$ from $K^s$ to $K$. Next, we develop an efficient gibbs sampling algorithm for this particular choice.

We sample a configuration for a clique $\mathcal{C}_{d,g}$ from its posterior $p(\mathcal{C}_{d,g}|W, Z_{\backslash \mathcal{C}_{d,g}})$. Since $\mathcal{C}_{d,g}$ can only take $K$ possible configuration, we use $\mathcal{C}_{d,g} = k$ to indicate that all variables in clique $\mathcal{C}_{d,g}$ taking value $k$. We show that:

$$p(\mathcal{C}_{d,g} = k|W, Z_{\backslash \mathcal{C}_{d,g}}) \propto \prod_{j=1}^{W_{d,g}} \left(\alpha_k + \mathcal{N}_{d,k\backslash C_{d,g}} + j - 1\right) \frac{\left(\beta_{w_{d,g,j}} + \mathcal{N}_{w_{d,g,j},k\backslash \mathcal{C}_{d,g}}\right)}{\left(\sum_{x=1}^{V} \beta_x + \mathcal{N}_{k\backslash \mathcal{C}_{d,g}} + j - 1\right)} \tag{4.10}$$

This can be derived as follows:

First, from equation 4.8, we have

$$P(Z, W) = \frac{1}{C} P_{\text{LDA}}(Z, W) \prod_{d,g} f(\mathcal{C}_{d,g}) \tag{4.11}$$

$$\propto \prod_{k=1}^{K} \left( \prod_{d=1}^{D} \Gamma(\alpha_k + \mathcal{N}_{d,k}) \frac{\prod_{x=1}^{V} \Gamma(\beta_x + \mathcal{N}_{x,k})}{\Gamma(\sum_{x=1}^{V} \beta_x + \mathcal{N}_k)} \right) \tag{4.12}$$

where the derivation of second line can be found in [100].

Second,

$$p(\mathcal{C}_{d,g} = k|W, Z_{\backslash \mathcal{C}_{d,g}}) \propto p(Z, W) \propto \frac{\Gamma(\alpha_k + \mathcal{N}_{d,k\backslash \mathcal{C}_{d,g}} + W_{d,g})}{\Gamma(\alpha_k + \mathcal{N}_{d,k\backslash \mathcal{C}_{d,g}})} * \tag{4.13}$$

$$\prod_{j=1}^{W_{d,g}} \frac{\Gamma(\beta_{w_{d,g,j}} + \mathcal{N}_{w_{d,g,j},k\backslash \mathcal{C}_{d,g}} + 1)}{\Gamma(\sum_{x=1}^{V} \beta_x + \mathcal{N}_{k\backslash \mathcal{C}_{d,g}} + W_{d,g})} \Big/ \frac{\Gamma(\beta_{w_{d,g,j}} + \mathcal{N}_{w_{d,g,j},k\backslash \mathcal{C}_{d,g}})}{\Gamma(\sum_{x=1}^{V} \beta_x + \mathcal{N}_{k\backslash \mathcal{C}_{d,g}})}$$

$$= \prod_{j=1}^{W_{d,g}} (\alpha_k + \mathcal{N}_{d,k\backslash \mathcal{C}_{d,g}} + j - 1) \frac{(\beta_{w_{d,g,j}} + \mathcal{N}_{k,w_{d,g,j}\backslash \mathcal{C}_{d,g}})}{(\sum_{x=1}^{V} \beta_x + \mathcal{N}_{k\backslash \mathcal{C}_{d,g}} + j - 1)} \tag{4.14}$$

where we utilize the fact that $\Gamma(x + 1) = x\Gamma(x)$.

For a "legitimate" $Z$, where the variables in the same clique take the same value, $p(Z, W|\alpha, \beta) = \frac{1}{C} P_{\text{LDA}}(Z, W|\alpha, \beta)$, which shows we can adopt the same hyper-parameter$(\alpha, \beta)$ optimization techniques as in LDA. In the experiment, we use the fixed-point method proposed by [101].

### 4.6.4 Topic visualization

There is a large literature in ranking terms and phrases for effective topical visualization. One method for selecting representative phrases (label) for a topic can be to minimize Kullback-Leibler divergence between word distributions and maximizing mutual information between label phrases and the topic model[102]. Another method attempts to extend the list of potential labels using external sources, such as wikipedia, and rank based on augmented candidate pool [103]. Other methods provide a parameterized multi-faceted ranking function that allows for a more user-controlled ranking[83]. These methods all provide a suggested methodology for ranking phrases within a topic model and can be easily incorporated into ToPMine.

For a more simplistic ranking function, we generalize the concept of N-most-probable terms in unigram LDA to our phrase output. By adopting the potential function given in equation 4.9, all variables in a clique are guaranteed to have the same latent topic. Since a clique corresponds to a phrase, naturally we assign the phrase to the same topic as shared by its constituents.

We utilize the topic assignment for each token from the last iteration of gibbs sampling, and define topical frequency (TF) for a phrase *phr* in topic k as the number of times it is assigned to topic k:

$$\text{TF}(phr,\text{k}) = \sum_{d,g} \mathcal{I}(\text{PI}_{d,g} == phr, C_{d,g} == k) \tag{4.15}$$

where $\mathcal{I}(\cdot)$ is the indicator function as used before, and $\text{PI}_{d,g}$ is the g$th$ phrase instance in d$th$ documents.

With this definition, we can visualize topic $k$ by sorting the phrases according to their topical frequency in topic $k$.

## 4.7 EXPERIMENTAL RESULTS

In this section, we start with the introduction of the datasets we used and methods for comparison. We then describe the evaluation on interpretability and scalability.

### 4.7.1 Datasets and methods for comparison

**Datasets**

We use the following six datasets for evaluation purpose:

- **DBLP titles**. We collect a set of titles of recently published computer science papers. The collection has 1.9M titles, 152K unique words, and 11M tokens.
- **20Conf**. Titles of papers published in 20 conferences related to the areas of Artificial Intelligence, Databases, Data Mining, Information Retrieval, Machine Learning, and Natural Language Processing - contains 44K titles, 5.5K unique words, and 351K tokens.
- **DBLP abstracts**. Computer science abstracts containing 529K abstracts, 186K unique words, and 39M tokens.
- **TREC AP news**. News dataset(1989) containing 106K full articles, 170K unique words, and 19M tokens.
- **ACL abstracts**. ACL abstracts containing 2k abstracts, 4K unique words and 231K tokens.
- **Yelp Reviews**. Yelp reviews containing 230k Yelp reviews and 11.8M tokens.

We perform stemming on the tokens in the corpus using the porter stemming algorithm[104] to address the various forms of words (e.g. cooking, cook, cooked) and phrase sparsity. We remove English stop words for the mining and topic modeling steps. Unstemming and reinsertion of stop words are performed post phrase-mining and topical discovery.

There are four directly comparable methods proposed in the literature: Turbo Topics, TNG, PD-LDA, and KERT.

### 4.7.2 Interpretability

We propose two user studies to demonstrate the effectiveness of our ToPMine framework.

#### Phrase Intrusion

First, we use an *intrusion detection* task which adopts the idea proposed by [79] to evaluate topical separation. The intrusion detection task involves a set of questions asking humans to discover the 'intruder' object from several options. Each question consists of 4 phrases; 3 of them are randomly chosen from the top 10 phrases of one topic and the remaining phrase is randomly chosen from the top phrases of a different topic. Annotators are asked to select the intruder phrase, or to indicate that they are unable to make a choice. The results of this task evaluate how well the phrases are separated in different topics

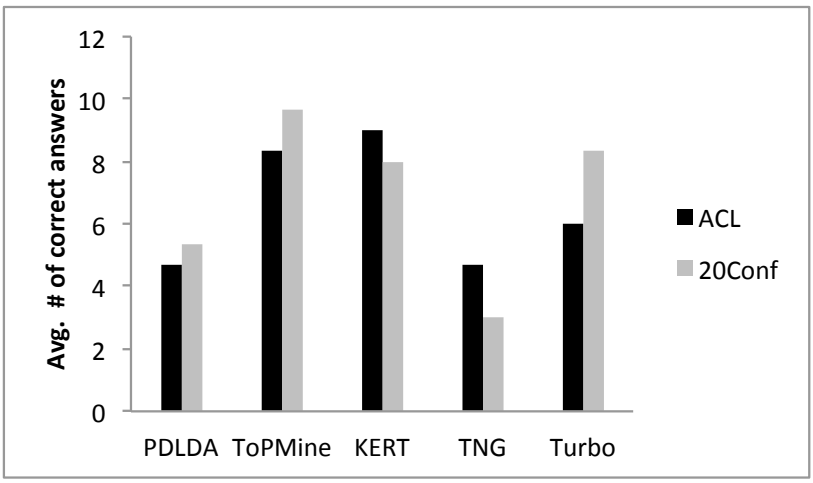



**Figure 4.3:** Phrase intrusion task. Test subjects were asked to identify an intruder phrase in a topic.

For each method, we sampled 20 Phrase Intrusion questions, and asked three annotators to answer each question. We report the average number of questions that is answered 'correctly' (matching the method) in Figure 4.3.

Domain Expert Evaluation

The second task is motivated by our desire to extract high-quality topical phrases and provide an interpretable visualization. This task evaluates both topical coherence on the full topical phrase list and phrase quality. We first visualize each algorithm's topics with lists of topical phrases sorted by topical frequency. For each dataset, five domain experts (computer science and linguistics graduate students) were asked to analyze each method's visualized topics and score each topical phrase list based on two qualitative properties:

- **Topical coherence:** We define topical coherence as homogeneity of a topical phrase list's thematic structure. This homogeneity is necessary for interpretability. We ask domain experts to rate the coherence of each topical phrase list on a scale of 1 *to* 10.
- **Phrase quality:** To ensure that the phrases extracted are meaningful and not just an agglomeration of words assigned to the same topic, domain experts are asked to rate the quality of phrases in each topic from 1 *to* 10.

For each expert, ratings were standardized to a z-score. We compute each algorithm's topical scores by averaging that of five experts. The results are shown in Figure 4.4 and Figure 4.5.

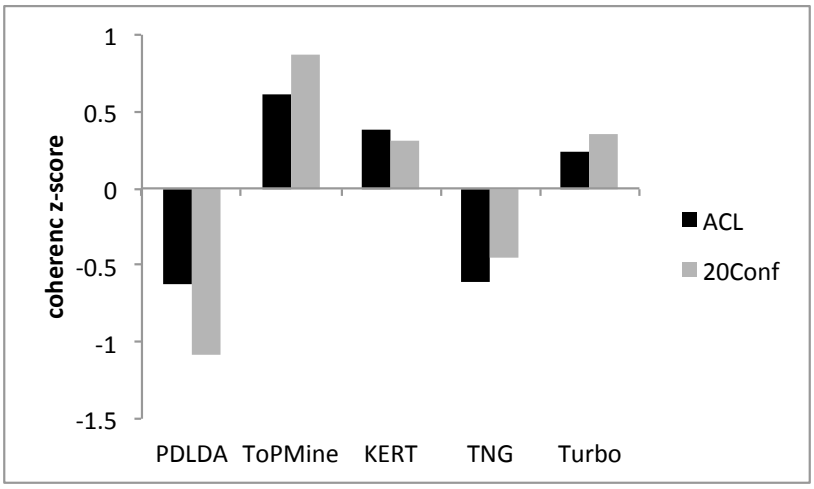



**Figure 4.4:** Coherence of topics. Domain experts were asked to rate the 'coherence' of each topic for each algorithm. Results were normalized into z-scores and averaged.

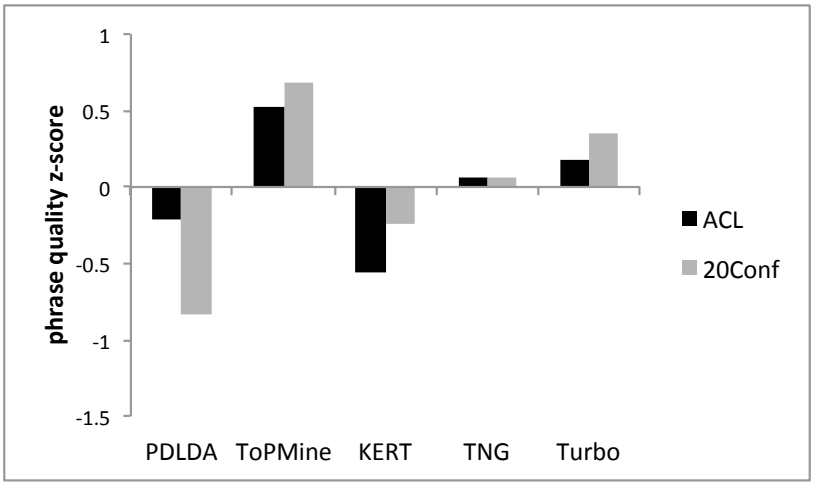



**Figure 4.5:** Phrase quality. Domain experts were asked to rate the quality of phrases for each topic for each algorithm. Results were normalized into z-scores and averaged.

Discussion of Userstudy

From Figures 4.3 and 4.4 we can tell that TopMine achieves similar performance to KERT in phrase intrusion, and demonstrates the best performance in topical coherence and phrase quality. We hypothesize that KERT's performance in phrase intrusion stems from its use of unconstrained frequent pattern mining and biased rankings towards longer phrases. Visual inspection suggests that many key topical unigrams are appended to common phrases, strengthening the notion of topical separation for all phrases. While this may aid KERT in phrase intrusion, we believe such practice lends to poor phrase quality, which is confirmed in Figure 4.5 as KERT demonstrates the lowest phrase-quality of the methods evaluated. A surprising occurrence is TNG and PD-LDA's poor performance in phrase intrusion. We

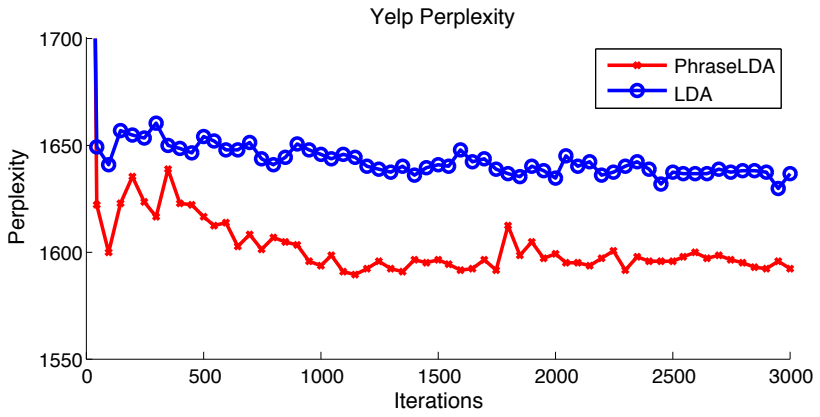



**Figure 4.6:** Yelp Reviews. A comparison of the perplexity of LDA vs PhraseLDA during Gibbs sampling inference.

suspect that this may be due to the many hyperparameters these complex models rely on and the difficulty in tuning them. In fact, the authors of PD-LDA make note that two of their parameters have no intuitive interpretation. Finally, Turbo Topics demonstrates above average performance on both datasets and user studies; this is likely a product of the rigorous permutation test the method employs to identify key topical phrases.

### 4.7.3 Perplexity

In addition to extracting meaningful and interpretable topical phrases, our ToPMine framework's PhraseLDA induces a statistical unigram topic model, upon the input corpus. To evaluate how well PhraseLDA's inference assumption that all words in our mined phrases should with high probability belong to the same topic, we evaluate how well the learned topic model predicts a held-out portion of our corpus. Because the generative process for PhraseLDA and LDA are the same, we can directly compare the perplexity between the two models to evaluate our method's performance.

As we can see on the Yelp reviews dataset in Figure 4.6, PhraseLDA performs significantly better than LDA demonstrating 45 bits lower perplexity than LDA. On the DBLP abstracts dataset, PhraseLDA demonstrates comparable perplexity to LDA. These results seem to validate the assumption that all words in our mined phrases should with high probability lie in the same topic. In addition, because our PhraseLDA can be seen as a more constrained version of LDA, these results provide an indication that our phrase mining method yields high-quality phrases as the perplexity of our learned model incorporating these phrases as constraints yields similar performance to LDA.

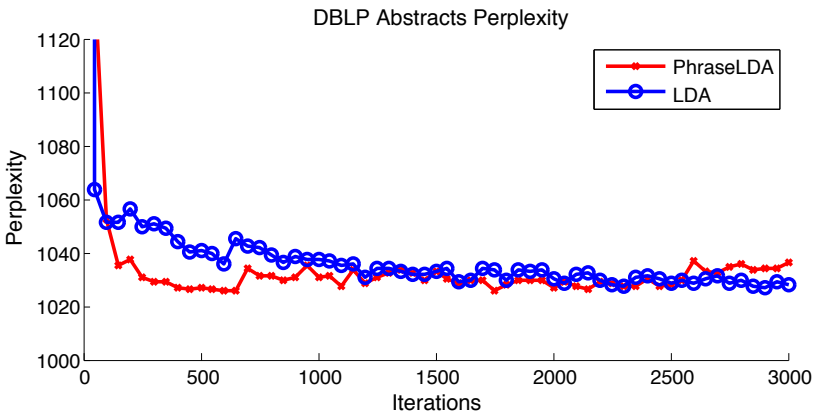



**Figure 4.7:** DBLP Abstracts. A comparison of the perplexity of LDA vs PhraseLDA during Gibbs sampling inference.

### 4.7.4 Scalability

To understand the run-time complexity of our framework, we first analyze the decomposition of ToPMine's runtime. On a high-level, ToPMine can be decomposed into two main separate procedures. The framework first involves frequent contiguous pattern mining followed by significant phrase construction. The second step is to take the 'bag-of-phrases' output as constraints in PhraseLDA. By separately timing these two steps in our framework, we can empirically analyze the expected runtime of each step. Figure 4.8 demonstrates the disparity in runtime between the phrase mining and topic modeling portions of ToPMine. Displayed on a log-scale for ease of interpretation we see that the runtime of our algorithm seems to scale linearly as we increase the number of documents (abstracts from our DBLP dataset). In addition, one can easily note that the phrase mining portion is of negligible runtime when compared to the topic modeling portion of the algorithm.

To evaluate our method's scalability to other methods, we compute our framework's runtime (on the same hardware) for datasets of various sizes and domains and compare them to runtimes of other state-of-the-art methods. For some datasets, competing methods could not be evaluated due to computational complexity leading to intractable runtimes or due to large memory requirements. We have attempted to estimate the runtime based on a smaller number of iterations whenever we face computational intractability of an algorithm on a specific dataset. We used an optimized Java implementation MALLET[105] for the TNG implementation and the topic modeling portions of KERT and Turbo Topics. For PD-LDA, we used the author's original C++ code. For LDA and PhraseLDA, the same JAVA implementation of PhraseLDA is used (as LDA is a special case of PhraseLDA). Because all these methods use Gibbs sampling to perform inference, we set the number of iterations to 1000. While we use hyperparameter optimization for our qualitative user-study tests and

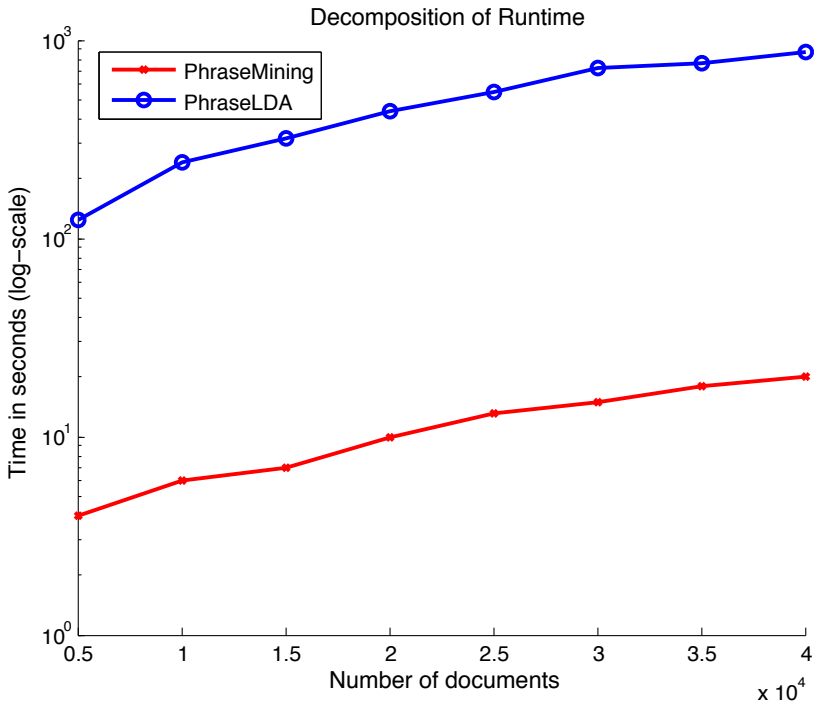



**Figure 4.8:** Decomposition of our topical phrase mining algorithm into its two components: phrase mining and phrase-constrained topic modeling. The plot above, which is displayed on a log-scale, demonstrates the speed of the phrase-mining portion. With 10 topics and 2000 Gibbs sampling iterations, the runtime of the topic modeling portion is consistently 40X the phrase mining.

perplexity calculations, we do not perform hyperparameter optimization in our timed test to ensure a fair runtime evaluation. The runtime for ToPMine is the **full runtime** of the framework including both phrase mining and topic modeling.

Table 4.3 shows the runtime of each method on our datasets. As expected, complex hierarchal models such as PD-LDA display intractable runtimes outside small datasets showing several magnitudes larger runtime than all methods except Turbo Topics. Turbo Topics displays a similar runtime due to the computationally intensive permutation tests on the back-off n-gram model. These methods were only able to run on the two sampled datasets and could not be applied to the full (larger) datasets. On short documents such as titles, KERT shows great scalability to large datasets barely adding any computational costs to LDA. Yet due to KERT's pattern-mining scheme, the memory constraints and the exponential number of patterns generated make large long-text datasets intractable. ToPMine is the only method capable of running on the full DBLP abstracts dataset with runtime in the same order as LDA. Under careful observation, PhraseLDA often runs in shorter time than LDA. This is because under our inference method, we sample a topic once for an entire multi-word phrase, while LDA samples a topic for each word.

Tables 4.4, 4.5, 4.6 are sample results of TopMine on three relatively large datasets - DBLP abstracts, AP News articles, and Yelp reviews. Our topical phrase framework was the only method capable on running on these three large, long-text datasets. In the visualization, we present the most probable unigrams from PhraseLDA as well as the most probable phrases below the unigrams. Automatic unstemming was performed as a post-processing step to visualize phrases in their most interpretable form. In many cases we see uninterpretable unigram topics that are made easier to interpret with the inclusion of topical phrases. Overall we can see that for datasets that naturally form topics such as events in the news and computer science subareas, ToPMine yields high quality topical phrases. For noisier datasets such as Yelp, we find coherent, yet lower quality topical phrases. We believe this may be due to the plethora of background words and phrases such as 'good', 'love', and 'great'. These and other words and phrases display sentiment and emphasis but are poor topical descriptors.

| *Method* | *sampled titles (k=5)* | *titles (k=30)* | *sampled abstracts* | *abstracts* |
|---|---|---|---|---|
| PDLDA | 3.72(hrs) | ∼20.44(days) | 1.12(days) | ∼95.9(days) |
| Turbo Topics | 6.68(hrs) | >30(days)* | >10(days)* | >50(days)* |
| TNG | 146(s) | 5.57 (hrs) | 853(s) | NA† |
| LDA | **65(s)** | 3.04 (hrs) | 353(s) | 13.84(hours) |
| KERT | 68(s) | 3.08(hrs) | 1215(s) | NA† |
| **ToPMine** | 67(s) | **2.45(hrs)** | **340(s)** | **10.88(hrs)** |

**Table 4.3:** We display the run-times of our algorithm on various datasets of different sizes from different domains. We sample 50 thousand dblp titles and 20 thousand dblp abstracts to provide datasets that the state-of-the art methods can perform on. For instances labeled *, we estimate runtime by calculating the runtime for one topic and extrapolating for k topics. For instances labeled ∼ we extrapolate by calculating runtime for a tractable number of iterations and extrapolating across all iterations. For instances labeled †, we could not apply the algorithm to the dataset because the algorithm exceeded memory constraints (greater than 40GB) during runtime.

## 4.8 DISCUSSION

In this chapter, we presented a topical phrase mining framework, ToPMine, that discovers arbitrary length topical phrases. Our framework mainly consists of two parts: phrase mining and phrase-constrained topic modeling. In the first part, we use frequent phrase mining to efficiently collect necessary aggregate statistics for our significance score - the objective function that guides our bottom-up phrase construction. Upon termination, our phrase mining step segments each document into a bag of phrases. The induced partitions are incorporated as constraints in PhraseLDA allowing for a principled assignment of latent topics to phrases.

| | *Topic 1* | *Topic 2* | *Topic 3* | *Topic 4* | *Topic 5* |
|---|---|---|---|---|---|
| 1-grams | problem | word | data | programming | data |
| | algorithm | language | method | language | patterns |
| | optimal | text | algorithm | code | mining |
| | solution | speech | learning | type | rules |
| | search | system | clustering | object | set |
| | solve | recognition | classification | implementation | event |
| | constraints | character | based | system | time |
| | programming | translation | features | compiler | association |
| | heuristic | sentences | proposed | java | stream |
| | genetic | grammar | classifier | data | large |
| n-grams | genetic algorithm | natural language | data sets | programming language | data mining |
| | optimization problem | speech recognition | support vector machine | source code | data sets |
| | solve this problem | language model | learning algorithm | object oriented | data streams |
| | optimal solution | natural language processing | machine learning | type system | association rules |
| | evolutionary algorithm | machine translation | feature selection | data structure | data collection |
| | local search | recognition system | paper we propose | program execution | time series |
| | search space | context free grammars | clustering algorithm | run time | data analysis |
| | optimization algorithm | sign language | decision tree | code generation | mining algorithms |
| | search algorithm | recognition rate | proposed method | object oriented programming | spatio temporal |
| | objective function | character recognition | training data | java programs | frequent itemsets |

**Table 4.4:** Five topics from a 50-topic run of ToPMine framework on our full DBLP abstracts dataset. Overall we see coherent topics and high-quality topical phrases we interpret as search/optimization, NLP, machine learning, programming languages, and data mining

| | *Topic 1* | *Topic 2* | *Topic 3* | *Topic 4* | *Topic 5* |
|---|---|---|---|---|---|
| 1-grams | plant | church | palestinian | bush | drug |
| | nuclear | catholic | israeli | house | aid |
| | environmental | religious | israel | senate | health |
| | energy | bishop | arab | year | hospital |
| | year | pope | plo | bill | medical |
| | waste | roman | army | president | patients |
| | department | jewish | reported | congress | research |
| | power | rev | west | tax | test |
| | state | john | bank | budget | study |
| | chemical | christian | state | committee | disease |
| n-grams | energy department | roman catholic | gaza strip | president bush | health care |
| | environmental protection agency | pope john paul | west bank | white house | medical center |
| | nuclear weapons | john paul | palestine liberation organization | bush administration | united states |
| | acid rain | catholic church | united states | house and senate | aids virus |
| | nuclear power plant | anti semitism | arab reports | members of congress | drug abuse |
| | hazardous waste | baptist church | prime minister | defense secretary | food and drug administration |
| | savannah river | united states | yitzhak shamir | capital gains tax | aids patient |
| | rocky flats | lutheran church | israel radio | pay raise | centers for disease control |
| | nuclear power | episcopal church | occupied territories | house members | heart disease |
| | natural gas | church members | occupied west bank | committee chairman | drug testing |

**Table 4.5:** Five topics from a 50-topic run of ToPMine on a large collection of AP News articles(1989). Overall we see high quality topical phrases and coherency of news topics such as environment, Christianity, Palestine/Israel conflict, Bush administration (senior), and health care

This separation of phrase-discovery from the topic model allows for less computational overhead than models that attempt to infer both phrases and topics and is a more principled approach than methods that construct phrases as a post-processing step to LDA. ToPMine demonstrates scalability on large datasets and interpretability in its extracted topical phrases beyond the current state-of-the-art methods.

Another note of discussion is that despite obtaining as input single and multi-word phrases, when computing a topic distribution, only distribution over words are generated. This not only makes perplexity comparisons between LDA and ToPMine valid, but provides a convenient case-study to assess the benefits of utilizing multiple granularity. Utilizing a distribution over phrases often leads to a data sparsity problem as phrases are less frequent

| | *Topic 1* | *Topic 2* | *Topic 3* | *Topic 4* | *Topic 5* |
|---|---|---|---|---|---|
| 1-grams | coffee | food | room | store | good |
| | ice | good | parking | shop | food |
| | cream | place | hotel | prices | place |
| | flavor | ordered | stay | find | burger |
| | egg | chicken | time | place | ordered |
| | chocolate | roll | nice | buy | fries |
| | breakfast | sushi | place | selection | chicken |
| | tea | restaurant | great | items | tacos |
| | cake | dish | area | love | cheese |
| | sweet | rice | pool | great | time |
| n-grams | ice cream | spring rolls | parking lot | grocery store | mexican food |
| | iced tea | food was good | front desk | great selection | chips and salsa |
| | french toast | fried rice | spring training | farmer's market | food was good |
| | hash browns | egg rolls | staying at the hotel | great prices | hot dog |
| | frozen yogurt | chinese food | dog park | parking lot | rice and beans |
| | eggs benedict | pad thai | room was clean | wal mart | sweet potato fries |
| | peanut butter | dim sum | pool area | shopping center | pretty good |
| | cup of coffee | thai food | great place | great place | carne asada |
| | iced coffee | pretty good | staff is friendly | prices are reasonable | mac and cheese |
| | scrambled eggs | lunch specials | free wifi | love this place | fish tacos |

**Table 4.6:** Five topics from a 10-topic run of our ToPMine framework on our full Yelp reviews dataset. Quality seems to be lower than the other datasets, yet one can still interpret the topics: breakfast/coffee, Asian/Chinese food, hotels, grocery stores, and Mexican food

than words. However our model leverages the restriction that phrases should belong to the same topic, but still decomposes each phrase into words when computing the topic distribution.

### 4.8.1 Future Works

One natural extension to this work is to extend our topic model PhraseLDA to use a nonparametric prior over topics. This will systematically allow for a data-driven estimate of the number of underlying topics in the corpus. Another area of work is in further scalability of the topic model portion. Currently the decreased computational complexity stems from the efficient phrase-mining. By investigating other methods for topical inference, the overall time complexity of ToPMine may be significantly reduced. Another area of focus is to address how the minimum support criterion and pruning strategies treat similar phrases as separate discrete structures, counting them separately. While this phenomenon doesn't affect the 'top-ranked' phrases, which have a count much larger than the minimum support, finding and merging similar phrases may lead to better recall and better topics. Further work may focus on strategies to identify and properly tie similar phrases. In Table 4.4 we notice background phrases like 'paper we propose' and 'proposed method' that occur in the topical representation due to their ubiquity in the corpus and should be filtered in a principled manner to enhance separation and coherence of topics.

Finally, further work should attempt to leverage subword information in the topic modeling perspective. While the current solution effectively utilizes word-level input and combines

them to form phrases, in many cases the words themselves may be infrequent and provide negligible input to the topic models. In these cases, to better model the content of each document, effectively decomposing these infrequent words into a subword-level granularity may allow for better topic identification for infrequent words. This will allow for the learned topic models to generalize to corpora with out-of-vocabulary words and allow for better learned topics.

# CHAPTER 5: MINING CROSS-LINGUAL PARALLEL SENTENCES FROM ALIGNED DOCUMENTS

In this chapter, we detail steps used to craft a large cross-lingual aligned document corpus. We assess the quality of this corpus and leverage cross-lingual sentence embeddings to mine parallel sentences.

Finally, we demonstrate how decomposing documents into sentences, embedding individual sentences, and then composing these embedded sentences can yield better document representations vs direct embedding of documents. We experimentally verify that constructing document representation from base sentence representations performs better document alignment.

## 5.1 INTRODUCTION

Cross-lingual document alignment aims to pair documents such that they are translations or near translations of each other. There are a variety of tasks in natural language processing that consume parallel cross-lingual data. Traditionally, machine translation approaches have leveraged parallel sentences as training data for use with sequence-to-sequence models. Other tasks include cross-lingual information retrieval and cross-lingual document classification. Additionally, cross-lingual data facilitates training cross-lingual representations such as multilingual BERT and XLM which are used in many NLP tasks[106, 107]. The availability of high-quality datasets is necessary to both train and evaluate models across these many tasks.

While it is possible to manually label aligned documents across languages, the process is costly and time consuming due to the quadratic search space for document pairs. Additionally, for low resource languages, identifying these cross-lingual document pairs is more difficult due to their relative scarcity. Furthermore, lack of access to qualified human annotators makes it necessary to have additional quality control in low-resource scenarios [108].

In this paper, we investigate whether we can rely on weak supervision to generate labels for document pairs. In particular, we focus on the weak signals embedded in the URLs of web documents, that can be used to identify the different translations of a single document across many languages. We propose a set of high-precision hand-crafted rules to automatically label a massive collection of 13 billion web documents and identify 54 million cross-lingual parallel documents in 92 language pairs. We evaluate the quality of our automatic-annotation setup using two approaches: (1) by running manual evaluation on a diverse sample of positively-labeled documents across nine languages; and (2) by leveraging the mined documents as

training data for a downstream machine translation task.

Finally, we also introduce a simple baseline that effectively aligns cross-lingual document pairs using solely textual content and in the presence of detractor documents which may not have any parallel counterpart. We release the dataset consisting of pairs of translated documents represented by URLs extracted from a massive collection web crawls. We hope that the size, diversity, and quality of this dataset spurs its use not only as a benchmark for document alignment, but also as supervision for a variety of cross-lingual tasks.

## 5.2 RELATED WORKS

The concept of crawling and mining the web to identify sources of parallel data has been previously explored [109]. A large body of this work has focused on identifying parallel text from multilingual data obtained from a single source: for example the United Nations General Assembly Resolutions or European Parliament parallel corpus [110, 111, 112]. These parallel corpora were curated from specific, homogeneous sources by examining the content and deriving domain-specific rules for aligning documents.

Other approaches have identified parallel documents in unstructured web corpora by relying on metadata. Some of these methods have focused on publication date and other temporal heuristics to aid in identifying parallel documents [113, 114, 115, 116, 117]. However, temporal features can be sparse, noisy, and unreliable. A different class of alignment methods rely on document structure [118, 119].

In the WMT-2016 bilingual document alignment shared task, many techniques applied retrieval and matching on translated 5-grams to query, retrieve, and align documents [120, 121]. Similar methods for generating candidates by retrieving matches based on the least frequent bi-lingual 5-grams have been proposed with the insight that rare snippets are more informative [122]. Both of these candidates rely on high-quality translation systems to translate either the source or the target. Such models may not exist, especially for low-resource language directions. The application of alignment to a variety of languages was not explored in WMT-2016 which only considered English to French document alignment – a high-resource direction.

Recently, the use of neural embedding methods has been explored for bilingual alignment of text at the sentence and document level. Other works propose using hierarchical document embeddings, constructed from sentence embeddings, for bilingual document alignment [123].

## 5.3 DATASET CREATION AND DESCRIPTION

The Common Crawl corpus is a publicly available crawl of the web. With a new snapshot uploaded each month, and over 2 billion pages released in each snapshot, this data is a vast resource with content across a large number of domains and languages. Previous works have leveraged the data from Common Crawl for mining ngram counts to perform language modeling [124]. Other works have mined Common Crawl for bitexts for machine translation [125]. However, this mining was performed on a small scale. For our dataset, we use snapshots published in 2018 covering twelve snapshots from January to December which is vastly larger than previous works.

### 5.3.1 Preprocessing

Extracting the textual content of Common Crawl web documents is a relatively challenging task that involves removing all tables, pictures, hyperlinks, and formatting markup. As such, the first preprocessing step is to remove all HTML tags and boiler-plate markup.

After content cleaning, the next step in preprocessing the data is deduplication. While investigating combining many Common Crawl snapshots, we found duplicate URLs both within an individual snapshot and almost always across snapshots. As our data curation method relies on unique URLs for each web document, we apply a heuristic to ensure each URL appears once within the final cleaned data. The first step is to normalize each URL; we perform this by simply removing the protocol and host name (e.g., `https://www.aaa.com` → `aaa.com`). Upon normalization, for each URL that appears more than once, we select the instance that possesses the longest document content. This heuristic assumes that occasionally, content is (1) deleted and gets shorter or (2) is amended and gets longer. In this case, it is preferable to operate on the larger content. Starting from 12 Common Crawl snapshots with a raw document count of 35.7 billion documents, upon deduplication, the resultant corpus is approximately 13.3 billion web documents from 64.8 million distinct web domains – a 63% reduction from the raw corpus.

### 5.3.2 Language Identification

The next step in the pipeline is to tag each document with the dominant language identifier. We utilize FastText, a lightweight text classifier that has been trained to detect more than 170 languages [15]. Because mixed language content is common, and boiler plate can often add noise to language identification, language identification may incorrectly tag doc-

uments. In case boilerplate (often at the beginning of a document) is tagged incorrectly, performing language identification on different segments can mitigate the noise introduced by boiler plate and correctly identify the dominant language within a document. To address this, we perform ensembling by predicting the language of a number of contiguous subsets of the document content. Majority voting is then used to tag the document with the predominant predicted language.

### 5.3.3 Language ID URL Matching

To identify pairs of cross-lingual documents, we apply a high-precision, low recall heuristic to assess whether two URLs represent web pages that are translations of each other. This heuristic presumes that two URLs, with high probability, refer to pages that are translations of each other if both can be transformed into the same string after stripping language identifiers. To improve recall, we allow matches where only one of the pair of URLs contain a language identifier e.g., `https://url.com` would be a match to `https://fr-fr.url.com`. We further ensure that these matches are high-precision by verifying that the language identifier stripped from the URL reflects the language of the web document document as predicted by the language identifier.

| **Source URL** | **Target URL** |
|---|---|
| **eng.**aaa.com | aaa.com |
| aaa.com/**en-gb**/b | aaa.com/**zh-cn**/b |
| aaa.com/**English**/b | aaa.com/**Yoruba**/b |
| aaa.com/b/**en** | aaa.com/b/**vi** |
| aaa.com/b/ | **thai.**aaa.com/b/ |
| aaa.com/b**&lang=english** | aaa.com/b**&lang=arabic** |
| aaa.com/b**?lang=en** | aaa.com/b**?lang=fr** |
| aaa.com/b | aaa.com/b**?lang=1** |

**Table 5.1:** URL matching via language identifiers.

Table 5.1 shows a few examples of pairs of aligned URLs. Alignment is performed by normalizing each URL by stripping its present language identifiers. Extra care is taken to ensure relevant indicators such as */*, *&*, and *?* are stripped as well to ensure proper alignment between URLs. For reproducibility, we publish an explicit list of patterns used along with the code implementing the pattern matching in the repository alongside the dataset.

For simplicity of implementation and reducing the volume of aligned documents, we restrict the source URL to English documents and allow the target URL to vary among the 92

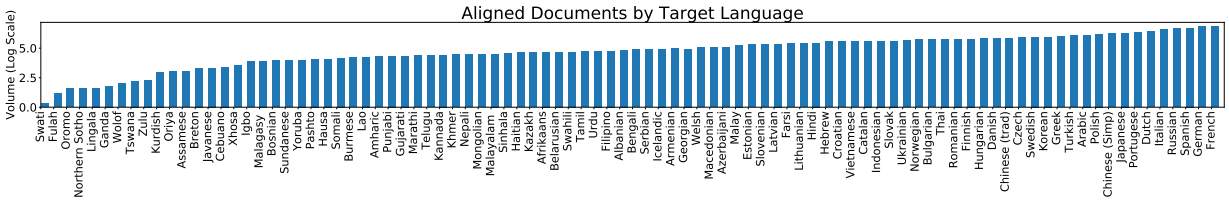



**Figure 5.1:** Per-language number of documents aligned to English documents.

target languages. Given these rules and restrictions, we mined 54 million aligned documents across 12 Common Crawl snapshots. See Figure 5.1 for detailed a breakdown per language. We assess the efficacy of this rule-based alignment in the next section.

We select a small subset of the original 12 Common Crawl snapshots to use for evaluating baseline document alignment methods. These 121K documents contain English and non-English documents from 450 web domains. Of these documents, 17.5K pairs are aligned as defined in our URL-aligned dataset. We release this test set as a tractable collection of documents on which to benchmark different alignment methods.

## 5.4 DATASET EVALUATION

In this section, we analyze the quality of our cross-lingual URL-aligned dataset. The first evaluation assesses the quality by measuring the precision of a representative sample of the URL-aligned data to human-annotated alignment judgments. The second evaluation assesses the data by utilizing the data in a downstream task. By first mining the aligned documents for parallel bitexts and using these bitexts as training data for massively multilingual machine translation, we can assess the overall quality of machine translation models trained solely from these mined bitexts.

### 5.4.1 Dataset Quality Evaluation

To assess the quality of the cross-lingual document pairs obtained by our method, we recruit human annotators to evaluate the alignments and provide an assessment of whether the documents in the pair are total or partial translations of each other. To perform the evaluation, we first selected six languages from various language families, scripts, and levels of resource availability. For each language, we randomly identified 30 pairs of URLs from different web domains aligned into English for a total of 180 pairs. To gather pairs from a diverse set of websites, each URL pair is selected from a distinct web domain.

Then, we tasked twelve human annotators to annotate URL pairs by loading the two web-

pages corresponding to each URL pair side by side and assessing whether or not the content rendered is both comparable *and* in the correctly tagged language. Each URL pair was evaluated by three human annotators to add a level of redundancy and measure annotator agreement. Note that the evaluation was performed in early November 2019, therefore for some of the pairs in the set, the document content had changed from the time when the Common Crawl snapshot was generated.

| | **Language** | $\mathbf{P}_{maj}$ | **K**$\boldsymbol{\alpha}$ | $\mathbf{P}_{adj}$ |
|---|---|---|---|---|
| **High** | German | 90.0 | 0.74 | 96.7 |
| | Chinese | 86.7 | 0.68 | 93.3 |
| **Mid** | Arabic | 83.3 | 0.72 | 90.0 |
| | Romanian | 76.7 | 0.50 | 96.7 |
| **Low** | Estonian | 83.3 | 0.68 | 90.0 |
| | Burmese | 86.7 | 0.88 | 100.0 |
| | **Avg** | 84.4 | 0.70 | 94.5 |

**Table 5.2:** Human evaluation of documents of different languages aligned to English. Languages are classified as high, medium or low resource based on the amount of mined documents. We report the majority-vote precision $\mathbf{P}_{maj}$ and the precision after accounting for experimental error $\mathbf{P}_{adj}$. Additionally, we report **K**$\boldsymbol{\alpha}$, a measure of inter-rater agreement.

In Table 5.2, we report the precision of our method to generate URL-aligned documents. As individual raters may have differing opinions on what constitutes a cross-lingual comparable document, we report results according to the majority vote. In addition, we report the inter-rating agreement among annotators as measured by the Krippendorff Alpha of the annotations [126]. After observing annotator comments and performing a round of error analysis on the pairs identified as misaligned, we identified the following reasons: (1) In 40% of the cases, the content of the rendered web-page has changed since the Common Crawl snapshot was generated or the URL redirects the user to a different page, while the original Common Crawl is a parallel document; (2) In 20% of cases, the content in one of the parallel documents appears to be much shorter than the document in the original (dominant) language but the message is the same, which many annotators didn't consider the document pairs as translations of each other; (3) In 10% of cases the majority of dynamic content within a document pair appears to be in the same language and only boilerplate text such as columns and title are translations; and the remaining 30% are truly non-comparable documents due to a myriad of different reasons. To alleviate the issues introduced by (1) due to experimental setup (i.e. using a freshly rendered web-page) and (2) due to guidelines issues (i.e. partial translations), we sent those cases for an additional round of annotation. The

resulting *adjusted* precision after the second round is observed as $P_{adj}$.

Overall, we observe that the URL pairs in Common Crawl appear to adhere to human-standards of comparability with a majority of measured directions achieving precision of over 90%.

### 5.4.2 Machine Translation Evaluation

Cross-lingual document pairs can be used to extract translations that are used for downstream training of sequence-to-sequence translation systems. Given a good pair of documents, our expectation is that a reasonable mining algorithm would be able to mine high quality translations, while the opposite is true for a low-quality pair of documents. In the WMT Parallel Corpus Filtering tasks the downstream performance on a translation task is used as a proxy to determine the quality of a similarity (or filtering) function [127, 128]. To assess the quality of the aligned document corpus, we propose a downstream task that leverages the aligned document data as a source of supervision for a massively multilingual machine translation task. If successful, our approach should be able to mine parallel sentences that are of comparable quality to recent approaches that leverage Wikipedia, a reliable source of comparable documents [129].

The first step is to decompose and mine the aligned document corpus for parallel sentences. We segment each document into sentences, then apply the Moses tokenizer (without true casing) to tokenize each sentence [130]. Given each document pair's decomposition into tokenized sentences, we seek to align sentences within each pair of documents. We can then aggregate these parallel sentences across all document pairs to form a parallel sentences dataset suitable for training machine translation models.

We apply a recent approach for mining parallel cross-lingual texts based on a distance measure in a joint multilingual sentence embedding space [131]. This method has been shown to accurately align and filter sentences for across a variety of low, mid, and high-resource directions [129, 132]. We apply the open-source LASER toolkit which provides a language agnostic sentence encoder and use the margin-based filtering criterion [133].

After mining parallel sentences from the aligned documents, we perform large-scale neural machine translation training on the extracted bitexts. First the data is processed to induce a 5000 subword vocabulary using SentencePiece [134]. The model used is a transformer model from fairseq with embeddings shared in the encoder and decoder, 5 encoder and decoder layers with dimensionality 512 are used, encoder and decoder FFN with 2 attention heads each with an embedding dimension of 2048 are used along with encoder and decoder normalization [135]. Dropout of 0.4, attention dropout of 0.2 and relu dropout of 0.2 are

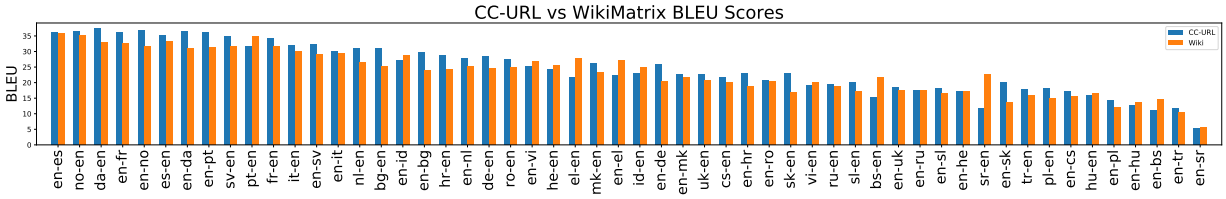



**Figure 5.2:** NMT performance on comparable directions of Wikipedia mined vs Common Crawl mined bitexts.

| | BLEU | | | | BLEU | | | | BLEU | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Lang** | **En–*x*** | ***x*–En** | **Vol** | **Lang** | **En–*x*** | ***x*–En** | **Vol** | **Lang** | **En–*x*** | ***x*–En** | **Vol** |
| French | 36.2 | 34.2 | 43.2 | Romanian | 20.6 | 27.6 | 4.3 | Estonian | 14.0 | 17.4 | 1.6 |
| Spanish | 36.3 | 35.3 | 40.3 | Vietnamese | 25.2 | 19.2 | 3.9 | Bengali | 15.0 | 11.0 | 1.0 |
| Russian | 17.5 | 19.6 | 29.8 | Ukrainian | 18.5 | 22.5 | 3.6 | Albanian | 21.3 | 28.9 | 1.0 |
| German | 25.8 | 28.3 | 26.7 | Greek | 22.4 | 21.8 | 3.6 | Macedonian | 22.6 | 26.3 | 0.7 |
| Italian | 30.0 | 32.0 | 24.6 | Korean | 11.4 | 5.8 | 3.6 | Urdu | 10.9 | 13.5 | 0.6 |
| Portuguese | 36.0 | 31.5 | 18.6 | Arabic | 10.3 | 18.8 | 3.4 | Serbian | 5.2 | 11.6 | 0.6 |
| Dutch | 27.8 | 31.0 | 15.5 | Croatian | 22.9 | 28.6 | 3.3 | Azerbaijani | 5.6 | 9.4 | 0.5 |
| Indonesian | 27.2 | 23.1 | 11.2 | Slovak | 20.0 | 23.1 | 3.2 | Armenian | 13.2 | 17.9 | 0.4 |
| Polish | 14.4 | 18.0 | 11.0 | Thai | 12.4 | 13.9 | 2.9 | Belarussian | 17.1 | 19.2 | 0.4 |
| Turkish | 11.6 | 17.7 | 7.5 | Hebrew | 17.1 | 24.4 | 2.6 | Georgian | 8.6 | 13.4 | 0.4 |
| Swedish | 32.3 | 34.9 | 6.1 | Hindi | 24.1 | 22.2 | 2.5 | Tamil | 16.8 | 6.4 | 0.3 |
| Danish | 36.6 | 37.3 | 5.0 | Hungarian | 12.6 | 15.8 | 2.5 | Marathi | 10.0 | 6.5 | 0.3 |
| Czech | 17.1 | 21.8 | 4.9 | Lithuanian | 15.1 | 20.4 | 2.1 | Kazakh | 3.4 | 6.2 | 0.2 |
| Bulgarian | 29.6 | 30.9 | 4.6 | Slovenian | 18.3 | 20.2 | 1.9 | Mongolian | 3.6 | 4.4 | 0.2 |
| Finnish | 11.9 | 16.7 | 4.6 | Persian | 9.8 | 16.4 | 1.7 | Burmese | 8.0 | 4.6 | 0.1 |
| Norwegian | 36.8 | 36.6 | 4.3 | | | | | Bosnian | 11.1 | 15.2 | 0.1 |

**Table 5.3:** BLEU scores of NMT models trained on bitext data mined from aligned Common Crawl documents evaluated on TED Talk test sets. Volume given as number of distinct aligned sentence pairs in millions.

applied. The adam optimizer is used to train the model for 100 epochs by optimizing a smoothed-cross entropy with 0.2 label smoothing.

After training models for each direction, we then evaluate the quality of the learned NMT models on a publicly available data set consisting of transcribed and translated TED talks in 50 languages [136]. This helps assess the quality of aligned data from our corpus as all parallel sentences must be extracted from aligned document pairs. Since the development and test sets were already tokenized, we first detokenize them using the Moses de-tokenizer.

In Tables 5.3 we report the BLEU scores from the mined bitexts from aligned documents on the TED talk dataset as well the number of distinct aligned sentence pairs (reported in millions). Based on these results, it appears that European languages yield higher quality sentences than non-European regardless of the resource level of the direction. Additionally, documents aligned across high-resource directions yield enough high-quality aligned data to learn high-quality models. While these BLEU scores should be taken in context of the volume of aligned bitexts, one can get an intuition as to the quality of the underlying URL-aligned documents the sentences were mined from from the resultant test-set BLEU scores.

Finally, we compare the test set BLEU scores to a dataset mined from Wikipedia using LASER sentence embedding and margin-based sentence alignment [129]. All preprocessing and experimental conditions including model hyper-parameters between these two NMT experiments were held constant making the BLEU scores directly comparable. As seen in Figure 5.2, sentences mined from the URL-aligned Common Crawl corpus is of comparable quality to the Wikipedia-mined data resulting in higher BLEU scores for 39 out of the 54 evaluated language directions (72.2%). This demonstrate that although we restrict parallel sentence alignment to documents that have been aligned by our URL rule-set, the mined sentences yielded are of high quality indicating adequately aligned documents.

## 5.5 DOCUMENT ALIGNMENT BASELINES & EVALUATION

In Section 5.4, we verify the quality of the URL-aligned dataset through human-evaluation and evaluation in a downstream task. In this section, we treat the URL-aligned dataset as a high-precision, low-recall dataset and evaluate baselines that score document pairs based on content rather than URL information. The scored document pairs are then aligned via a greedy bipartite matching algorithm. The resulting alignments are evaluated on a subset of the URL-aligned dataset which is treated as ground truth.

### 5.5.1 Problem Definition

Given a set of source documents, $D_s$ and a set of target documents $D_t$, cross-lingual document alignment aims to find the largest set of pairs of documents from source to target $(d_s, d_t)$ where $d_s \in D_s$ and $d_t \in D_t$ such that each source document and target document can only be used in at most a single pair.

To find the best possible mapping between $D_s$ and $D_t$ we require two components: 1) a similarity function $\phi(d_s, d_t)$ which is used to score a set of candidate documents according to their relatedness; and 2) an alignment or matching algorithm which uses the scores for each of the pairs in $D_s \times D_t$ to produce an alignment of size $min(|D_s|, |D_t|)$ representing the best mapping according to $\phi(d_s, d_t)$.

In the remainder of this section, we introduce our proposed baseline document pair similarity functions and a simple matching algorithm that aligns source and target documents.

### 5.5.2 Embedding-Based Document Similarity

To guide the alignment algorithm, a notion of cross-lingual document similarity is necessary. This score should capture the fact that two documents are *semantically* similar despite having some or all of their content in different languages. We describe two simple language-agnostic document embedding methods. These embeddings leverage LASER [137], a multilingual sentence representation that uses byte-pair encoding to share the same vocabulary among all languages and trained on parallel sentences covering 93 languages.

**Direct Embedding** The first baseline, Direct Embedding (DE) uses a standard cross-lingual encoder to directly embed each document. Each document $d$ has its dense vector representation $\mathbf{v}_d$ computed by applying the open-source cross-lingual LASER encoder to its *full textual content*.

**Sentence Average Embedding** The second baseline, Sentence Average (SA), performs document embedding by first decomposing each document into smaller, semantically meaningful sentences, embedding each sentence, then combining these sentence representations. Given a document $d$, we segment it into a list of sentences $\{s_i\}_{i=1}^{n}$. This time, the LASER encoder is used to encode each sentence $s_i$ into a dense vector $\mathbf{v}_{s_i}$. After embedding each sentence in a document, document embedding is performed by averaging these sentence vectors into a document vector $\mathbf{v}_d$ as follows:

$$\mathbf{v}_d = \frac{1}{n} \sum_{i=1}^{n} \mathbf{v}_{s_i} \tag{5.1}$$

**Scoring** Using the dense document representations for each document from the source and target sets, the next step is to score pairs to evaluate how semantically similar documents are. Given two documents $a$ and $b$, We compute their semantic similarity using a cosine similarity score:

$$sim(a, b) = \frac{\mathbf{v}_a \cdot \mathbf{v}_b}{||\mathbf{v}_a||\ ||\mathbf{v}_b||} \tag{5.2}$$

### 5.5.3 Competitive Matching Alignment

Using the baseline scoring function, we score all document pairs in the same web domain that belong to the source and target languages respectively. As such, for any given domain,

each document in the source document set, $D_s$ is paired with each document in the target set, $D_t$, yielding $D_s \times D_t$ scored pairs – a fully connected bipartite graph. Just like in [138], the expected output assumes that each page in the non-dominant language has a translated or comparable counterpart. This yields a $min(|D_s|, |D_t|)$ expected number of aligned pairs.

While an optimal matching maximizing scoring can be solved using the Hungarian algorithm, the complexity of this algorithm is $\mathcal{O}(max(|D_s||D_t|)^3)$ which is intractable to even moderately sized web domains [139]. As such, similar to previous work where a one-to-one matching between English and non-English documents is enforced by applying, competitive matching, a greedy bipartite matching algorithm [138].

**Algorithm 5.1:** Competitive Matching

**Input:** $P = \{(d_s, d_t) | d_s \in D_s, d_t \in D_t\}$
**Output:** $P' = \{(d_{s,i}, d_{t,i}), ...\} \subset P$

```
scored ← {(p, score(p)) for p ∈ P}
sorted ← sort(scored) in descending order
aligned ← ∅
S_s ← ∅
S_t ← ∅
for d_s, d_t ∈ sorted do
    if d_s ∉ S_s ∧ d_t ∉ S_t  aligned ← aligned ∪ {(d_s, d_t)}
    S_s ← S_s ∪ d_s
    S_t ← S_t ∪ d_t
end
return aligned
```

In Algorithm 5.1, the algorithm first scores each candidate document pair using the document similarity scoring function. These candidates are then sorted in order of most similar to least similar using their numerical score. The algorithm then iteratively chooses a document pair with the highest score as long as the $d_s$ and $d_t$ of each pair have not been used in a previous (higher scoring) pair. The algorithm terminates when $min(|D_s|, |D_t|)$ pairs have been selected. Unlike the Hungarian algorithm, the runtime complexity is a more tractable $\mathcal{O}(|D_s||D_t| \times \log(|D_s||D_t|))$ which is dominated by the cost of sorting all candidate pairs.

### 5.5.4 Baseline Results

We evaluate the baseline scoring by aligning the documents from a subset of the 12 Common Crawl snapshots. We score document pairs from the source and target languages within the same webdomain using the DE and SA embedding methods respectively and computing cosine similarity between the two dense representations. For the alignment, we report

| | Recall | | | | | Recall | | | | | Recall | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Lang** | **DE** | **SA** | **DE+** | **SA+** | **Lang** | **DE** | **SA** | **DE+** | **SA+** | **Lang** | **DE** | **SA** | **DE+** | **SA+** |
| French | 0.36 | **0.85** | 0.39 | **0.85** | Romanian | 0.10 | 0.39 | 0.15 | **0.40** | Estonian | 0.26 | 0.47 | 0.28 | **0.52** |
| Spanish | 0.34 | **0.59** | 0.34 | 0.53 | Vietnamese | 0.04 | 0.22 | 0.06 | **0.28** | Bengali | 0.02 | 0.27 | 0.05 | **0.32** |
| Russian | 0.03 | 0.43 | 0.06 | **0.64** | Ukrainian | 0.03 | 0.58 | 0.05 | **0.68** | Albanian | 0.15 | **0.59** | 0.23 | 0.56 |
| German | 0.38 | **0.74** | 0.52 | **0.74** | Greek | 0.02 | 0.22 | 0.05 | **0.31** | Macedonian | 0.00 | 0.29 | 0.02 | **0.33** |
| Italian | 0.17 | 0.46 | 0.22 | **0.47** | Korean | 0.01 | 0.25 | 0.06 | **0.34** | Urdu | 0.02 | 0.21 | 0.06 | **0.22** |
| Portuguese | 0.14 | 0.35 | 0.17 | **0.36** | Arabic | 0.02 | 0.28 | 0.04 | **0.32** | Serbian | 0.06 | **0.59** | 0.06 | **0.59** |
| Dutch | 0.25 | **0.50** | 0.28 | 0.49 | Croatian | 0.16 | **0.37** | 0.16 | **0.37** | Azerbaijani | 0.04 | 0.11 | 0.08 | **0.34** |
| Indonesian | 0.07 | **0.48** | 0.11 | 0.47 | Slovak | 0.24 | **0.43** | 0.20 | 0.41 | Armenian | 0.01 | 0.13 | 0.02 | **0.18** |
| Polish | 0.17 | **0.38** | 0.17 | **0.38** | Thai | 0.01 | **0.21** | 0.02 | 0.19 | Belarusian | 0.04 | 0.40 | 0.07 | **0.47** |
| Turkish | 0.09 | 0.34 | 0.12 | **0.38** | Hebrew | 0.01 | 0.09 | 0.05 | **0.18** | Georgian | 0.04 | 0.19 | 0.06 | **0.24** |
| Swedish | 0.12 | **0.42** | 0.19 | 0.40 | Hindi | 0.04 | 0.20 | 0.04 | **0.27** | Tamil | 0.02 | 0.15 | 0.02 | **0.20** |
| Danish | 0.19 | 0.61 | 0.27 | **0.62** | Hungarian | 0.15 | 0.43 | 0.15 | **0.49** | Marathi | 0.00 | 0.06 | 0.02 | **0.11** |
| Czech | 0.16 | 0.38 | 0.15 | **0.40** | Lithuanian | 0.07 | 0.69 | 0.11 | **0.73** | Kazakh | 0.03 | 0.22 | 0.05 | **0.31** |
| Bulgarian | 0.04 | 0.37 | 0.07 | **0.43** | Slovenian | 0.12 | 0.32 | 0.13 | **0.33** | Mongolian | 0.01 | 0.08 | 0.03 | **0.13** |
| Finnish | 0.06 | 0.38 | 0.06 | **0.47** | Persian | 0.03 | 0.24 | 0.06 | **0.32** | Burmese | 0.00 | 0.04 | 0.01 | **0.10** |
| Norwegian | 0.13 | **0.33** | 0.13 | **0.33** | | | | | | Bosnian | 0.15 | 0.63 | 0.18 | **0.64** |
| **AVG** | 0.17 | 0.48 | 0.20 | **0.50** | **AVG** | 0.07 | 0.33 | 0.09 | **0.37** | **AVG** | 0.05 | 0.28 | 0.08 | **0.33** |

**Table 5.4:** Recall from Common Crawl documents aligned using the baseline content-based alignment methods.

the performance for each embedding method without enforcing a 1-to-1 alignment between source and targets (DE & SA) and after applying our competitive matching alignment algorithm (DE+CM, SA+CM) as described in Algorithm 5.1.

When reporting results for DE & SA without applying competitive matching, we explicitly enforce that $min(|D_s|, |D_t|)$ document pairs are output by identifying the smaller between $D_s$ and $D_t$, then selecting the best matching document to pair with from the larger set for each document in the smaller set. This ensures that $min(|D_s|, |D_t|)$ pairs are generated making comparisons between methods using and not using competitive matching fair.

Recall (i.e. what percentage of the aligned pages in the test set are found) is computed on a test-set consisting of pairs from the URL-aligned documents, which we verified have high-precision and we treat as the ground-truth test set.

We show the alignment results in Table 5.4. Comparing DE which directly applies LASER to the entirety of the document content, we see that performance is significantly worse than SA which averages the individual sentence embeddings. We suspect this may be the case for two reasons (1) sentence encoders may suffer at representing the semantic meaning of long documents as the model is originally trained on sentences (2) there may be noisy boiler plate content at the beginning of each web document that is less useful semantically but dominates the representation.

Observing the effects of competitive matching (CM) on alignment, it appears to consistently improve the overall alignment recall for scoring using both DE and SA and across all levels of resource availability. We believe that this may be because it prevents many

sentences from being aligned to a single "hub" sentence. As such, the strongest baseline combines a sentence averaging approach for document representation for scoring, then subsequently selects the final aligned pairs using competitive matching.

From the results, we observe as the resource availability increases, our alignment baselines perform better. This may be due to the fact that the LASER embedding models were trained with parallel data and more high-resource parallel sentences were used for training. Finally, it appears that across low, mid, and high-resource directions, European languages appear to be consistently easier to align than non-European languages. For example, as seen in Table 5.4, Albanian, Serbian, Bosnian, and Belarusian were all aligned with over 0.45 recall despite being low-resource. We posit that this is a by-product of the shared semantic subword vocabulary used by LASER improving performance for low-resource European languages due to their similarity of script and linguistic similarities with the many high-resource European languages.

## 5.6 DISCUSSION

In this chapter, we apply URL-matching rules to curate a high-quality cross-lingual documents dataset from the commoncrawl corpus. Our dataset contains document pairs from 92 different languages aligned with English. We first directly evaluate the quality of the URL-aligned pairs using human annotators. We further evaluate the URL-aligned documents in a downstream machine translation task by decomposing the aligned documents into aligned sentences, and then training machine translation models across all 92 directions. Finally, we introduce and evaluate a new general embedding-based baseline technique for aligning documents based on content rather than meta-information like URLs. Our results indicate there is further work to be done to improve document alignment, especially for low-resource languages and that intelligent alignment schemes can significantly improve overall alignment performance across many language directions.

Some insights from these results is that the proper level of semantic granularity is crucial to the cross-lingual document alignment task. Attempting to directly embed the entire document into a cross-lingual space greatly under-performed compared to leveraging sentence embeddings. Further works should attempt to leverage these results to better construct document embeddings using hierarchical techniques as opposed to direct embeddings. Building up from subwords, words, and phrases can be used to learn better sentence representations. These sentence representations can be combined more intelligently to form document representations that can be used in a variety of downstream tasks from alignment, to retrieval, to classification.

# CHAPTER 6: CROSS-LINGUAL DOCUMENT ALIGNMENT WITH SENTENCE REPRESENTATIONS

In this chapter, we continue investigating the use of cross-lingual sentence embeddings for mining cross-lingual documents. As opposed to constructing document representations, we propose aligning documents by first decomposing each document into sentences, and directly comparing constituent sentences between documents. We develop a technique based on earth mover's distance that leverages the sentence embeddings to better model cross-lingual document semantic similarity. We demonstrate how this can be used to better identify cross-lingual document pairs especially on low-resource language pairs.

## 6.1 INTRODUCTION

While the World Wide Web provides a large amount of monolingual text, cross-lingual parallel data is more difficult to obtain. Despite its scarcity, parallel cross-lingual data plays a crucial role in a variety of tasks in natural language processing. Traditionally, machine translation approaches have leveraged parallel sentences as training data for use with sequence-to-sequence models. Previous works have also shown that training on sentences extracted from parallel or comparable documents mined from the Web can improve machine translation models [113]. Parallel cross-lingual documents can also be used for learning word-level translation lexicons [140, 141]. Other tasks that leverage these parallel data include cross-lingual information retrieval and document classification. Additionally, cross-lingual data facilitates training multilingual representations such as XLM [107] which can be used as input to many downstream NLP tasks yielding language-agnostic NLP.

Document alignment is a method for obtaining cross-lingual parallel data that seeks to pair documents in different languages such that pairs are translations or near translations of each other. As seen in Figure 6.1, this involves a one-to-one pairing of documents in a source language with documents in a target language. While it is possible to manually align documents across languages, the process is costly and time consuming due to the quadratic search space for document pairs. Additionally, for low resource languages, identifying these cross-lingual document pairs is difficult due to their relative scarcity and the scarcity of human annotators familiar with the languages.

To automate and scale the process of identifying these documents pairs, we introduce an approach to accurately mine comparable web documents across a variety of low, mid, and high-resource language directions. Previous approaches have been applied to homogeneous corpora, however mining the Web involves analyzing a variety of heterogeneous data

sources [142]. Other approaches rely on corpus-specific features such as metadata and publication date which can be inconsistent and unreliable [113, 117]. Related methods utilize document structure when calculating document similarity [118, 119]. However, when mining large, unstructured collections of web documents these features are often missing or unreliable. As such, we introduce an approach that aligns documents based solely on semantic distances between their textual content.

For our approach, we first decompose documents into sentences, and encode each sentence into a cross-lingual semantic space; this yields a bag-of-sentences representation for each document. Utilizing the dense, cross-lingual representation of sentences, we then formulate document similarity as a variant of earth mover's distance where the objective is to move probability mass from source-document sentences to target-document sentences. We then leverage these document distances as a guiding metric for identifying cross-lingual document pairs and demonstrate experimentally that our proposed method outperforms state-of-the-art baselines that utilize cross-lingual document representations.

## 6.2 RELATED WORKS

The concept of crawling and mining the web to identify sources of parallel data has been previously explored [109]. A large body of this work focuses on identifying parallel text from multilingual data obtained from a single source. For example, one parallel corpus was curated from the United Nations General Assembly Resolutions [110, 111]. Another parallel corpus was curated from documents from the European Parliament [112]. Both of these parallel corpora were curated from specific, homogeneous sources by examining the content and deriving domain-specific rules for aligning documents. As such, these techniques do not generalize to arbitrary web-domains obtained from large-scale web scraping efforts.

Other approaches have identified parallel documents in unstructured web corpora by relying on metadata. Some of these methods have focused on publication date and other temporal heuristics to aid in identifying parallel documents [113, 114, 115, 116, 117]. However, temporal features are often sparse, noisy, and unreliable. Another class of alignment methods rely on document structure [118, 119]. Once again these document structure features can be sparse in web-domains and may require hand-crafted rule-sets to fully leverage. These rule-sets may not generalize to new domains.

In the WMT-2016 bilingual document alignment shared task, many techniques were proposed to retrieve, score, and align cross-lingual document pairs [120]. However this shared task only considered English to French – a high-resource direction. The techniques were not evaluated on languages of varying resource availability and the proposed techniques were

not readily extendable to application on a massively multilingual scale.

Some of the proposed methods translated the target corpus into the source language, then applied standard retrieval and matching approaches on translated 5-grams to query, retrieve, and align documents [121]. Similar methods for generating candidates by retrieving matches based on the least frequent bi-lingual 5-grams were proposed with the insight that rare snippets are more informative and can better identify cross-lingual pairs [122]. Both of these methods rely on high-quality translation systems to translate either the source or the target, however such models may not exist, especially for low-resource language directions. Additionally, these methods leverage rare n-grams to identify likely candidates. However it is precisely low-frequency words and phrases that are likely to be mistranslated by machine translation systems.

In the shared task, many document similarity measures were investigated for use in aligning English to French web documents. One method utilized a phrase table from a phrase-based statistical machine translation system to compute coverage scores, based on the ratio of phrase pairs covered by a document pair [122]. Other methods utilize the translated content of the target (French) document, and find the source (English) corresponding document based on n-gram matches in conjunction with a heuristic document length ratio [121, 143]. Other methods translate the target documents into the source language and apply cosine similarity between tf/idf weighted vectors on unigrams and n-grams [138, 144, 145]. Finally, several methods were introduced that leverage metadata in each document such as links to documents, URLs, digits, and HTML structure [146, 147].

Recently, the use of neural embedding methods has been explored for bilingual alignment of text at the sentence and document level. One method proposes using hierarchical document embeddings, constructed from sentence embeddings, for bilingual document alignment [123]. Another method leverages a multilingual sentence encoder to embed individual sentences from each document, then performs a simple vector average across all sentence embeddings to form a dense document representation. Cosine similarity is then used to identify document pairs [6].

Word mover's distance (WMD) has been recently used for document similarity and classification [148, 149, 150]. However these methods have been solely applied in the monolingual space. Other methods have been proposed to leverage EMD for cross-lingual document retrieval [151], however these methods treat individual words as the base semantic unit for comparison. The large number of tokens present in web documents coupled with the cubic complexity of WMD make these approaches intractable for large-scale web-alignment.

Finally, sentence mover's similarity has been proposed for automatically evaluating machine-generated texts outperforming ROUGE [152]. However the proposed method is purely mono-

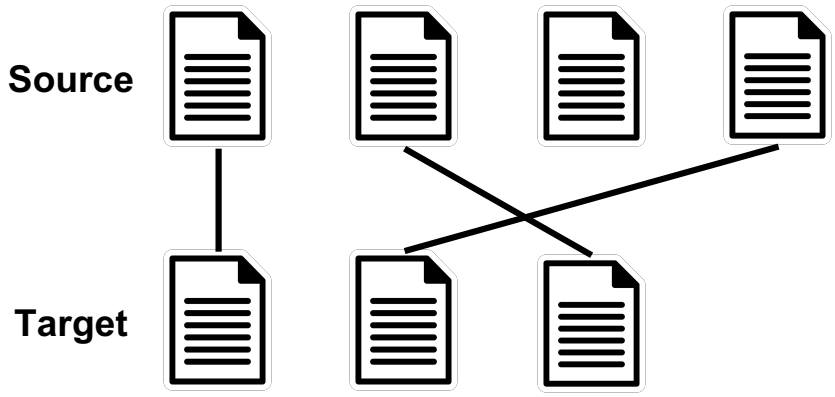



**Figure 6.1:** Documents in a source and target language in the same web-domain. Solid lines indicate cross-lingual document pairs.

lingual and sentence representations are constructed by summing individual word embeddings.

## 6.3 PROBLEM DEFINITION

Given a set of source documents, $D_s$ and a set of target documents $D_t$, there exist $|D_s| \times |D_t|$ potential pairs of documents where each document pair is of the form $(d_s, d_t)$ s.t. $d_s \in D_s$ and $d_t \in D_t$ respectively. Let $\mathcal{P}$ be the set of all candidate pairs $(D_s \times D_t)$. Then cross-lingual document alignment aims to find the largest mapping from source documents to target documents, $\mathcal{P}' \subset \mathcal{P}$, s.t. given an $D_s$ and $D_t$ where, without a loss of generality, $|D_s| \leq |D_t|$, the largest injective mapping between $D_s$ and $D_t$:

$$\forall a, b \in D_s, (a, c) \in \mathcal{P}' \wedge (b, c) \in \mathcal{P}' \implies a = b \tag{6.1}$$

In other words, each source document and target document can only be used in at most a single pair.

This can be seen in Figure 6.1 where within the same web-domain, documents can be separated into two disjoint sets: documents in the source language ($D_s$) and documents in the target language ($D_t$). The task then becomes to match each source document to a unique target document where possible.

To find the best possible mapping between $D_s$ and $D_t$ we require two components: 1) a similarity function $\phi(d_s, d_t)$ which is used to score a set of candidate document pairs according to their semantic relatedness; and 2) an alignment or matching algorithm which uses the scores for each of the pairs in $D_s \times D_t$ to produce an alignment of size $min(|D_s|, |D_t|)$ representing the best mapping according to $\phi(d_s, d_t)$.

The remainder of this paper is organized as follows: In Section 6.4 we introduce our proposed cross-lingual document distance metric and in Section 6.5 we describe a simple algorithm that leverages this metric to perform cross-lingual document alignment. In Section 6.7 we evaluate our method end-to-end and conduct ablation studies on different design decisions in Section 6.6. We conclude in Section 6.8.

## 6.4 CROSS-LINGUAL SENTENCE MOVER'S DISTANCE

WMD extends the notion of earth mover's distance, a measure of distance between two probability distributions over a metric space, to measure semantic document similarity. This adaptaion represents each document as a bag-of-words (BOW) normalized by their relative counts in the document, and measures distances between words using standard word embeddings such as Word2Vec or Glove [20, 153]. The distance can then be formulated as the minimum amount of distance that the embedded words of one document need to "travel" to reach the embedded words of another document.

While demonstrating powerful results in classification and retrieval tasks, WMD fails to generalize to our use case for two reasons: (1) the technique relies on monolingual word representations which fail to capture the semantic distances between documents whose content are in different languages and (2) web documents may be thousands of words long or even how no word boundaries in certain languages. As such, WMD becomes quickly intractable or infeasible on these web-documents.

To address this, we adapt WMD to better measure the similarity between two documents in potentially different languages. We perform this by introducing a distance metric we dub cross-lingual sentence mover's distance (XLSMD). We show that by representing each document as a bag-of-sentences (BOS) and leveraging recent improvements in cross-lingual sentence representations, XLSMD can better identify cross-lingual document pairs.

### 6.4.1 Multilingual Sentence Embeddings

Evaluating the distance between document pairs involves breaking up documents into constituent semantic units such as sentences and measuring the distance between these units. In order to evaluate the distance between documents composed in many different languages, we require a joint embedding scheme for all the considered languages.

Previous approaches have trained bi-lingual embeddings for each and every language pair under consideration [154, 155, 156]. However, training bilingual embedding models for each language pair is difficult to scale beyond a handful of language pairs. Instead, we adopt

the massively multilingual sentence representation proposed in the LASER toolkit [137]. Figure 6.2 demonstrates the training process for learning to encode sentences into a shared multilingual embedding space using a sequence-to-sequence model with a shared BPE vocabulary. This approach simultaneously models 93 languages covering 23 different alphabets into a joint embedding space. LASER accomplishes this by training a sequence-to-sequence system on many language pairs at once using a shared encoder and a shared byte-pair encoding (BPE) vocabulary for all languages. The sentence representation is obtained by max-pooling over all encoder output states [137].

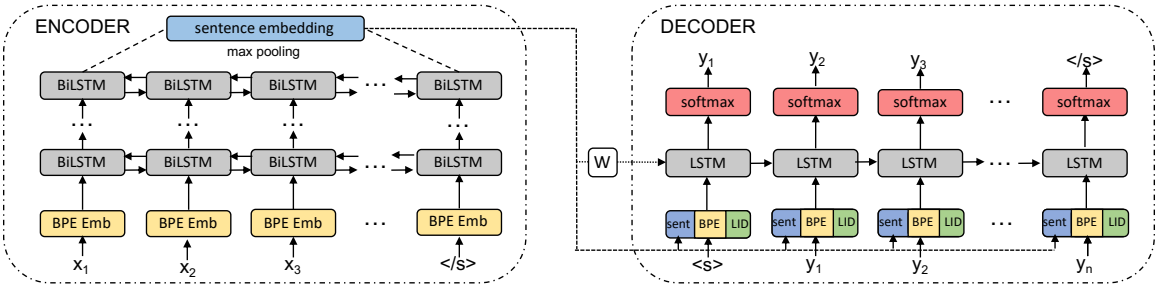



**Figure 6.2:** Architecture of the system used to train massively multilingual sentence embeddings [137].

For our XLSMD approach, we leverage these multilingual sentence embeddings to measure euclidean distance between sentences in the source document and target document.

### 6.4.2 Cross-Lingual Sentence Mover's Distance

Our proposed XLSMD solves the same optimization problem as WMD, but utilizes cross-lingual sentence embeddings instead of word embeddings as the base semantic unit of a document. In particular, we utilize LASER sentence representations [137] whereby each sentence is encoded using an LSTM encoder into a fixed-length dense representation as described in Section 6.4.1.

XLSMD is a distance metric based on the Wasserstein metric also known as the earth mover's distance (EMD) [157]. In our approach, we adapt the EMD to measure the distance between two documents by comparing the distributions of sentences within each document. This metric can be viewed as the sentence-based adaptation of WMD [148]. More specifically, XLSMD represents each document as a *bag-of-sentences* (BOS) where each sentence has associated with it some probability mass. Leveraging that distances can be computed between dense sentence embeddings, the overall document distance can then be computed by examining how close the distribution of sentences in the source document is to sentences

in the target document. This formulation captures note only item similarity on a BOS histogram representations of the text, but also the multilingual sentence embedding distances. We formulate that the distance an arbitrary pair of documents A and B is the minimum cost of transforming one document into the other.

For our basic formulation of XLSMD, each document is represented by the relative frequencies of sentences, i.e., for the $i_{th}$ sentence in the document,

$$d_{A,i} = \frac{count(i)}{\sum_{s \in A} count(s)} \tag{6.2}$$

where $\sum_{s \in A} count(s)$ is the total number of sentence in document A, and $\mathrm{d}_{B,i}$ is defined similarly for document B. Under this assumption, each individual sentence in a document is equally important and probability mass is allocated uniformly to each sentence. Later, we will investigate alternative schemes to allocating probability mass to sentences.

Now let the $i_{th}$ sentence be represented by a vector $v_i \in R^m$. This length-m dense embedding representation for each sentence allows us to define distances between the $\mathrm{i}_{th}$ and $\mathrm{j}_{th}$ sentences. We denote $\Delta(i, j)$ as the distance between the $i_{th}$ and $j_{th}$ sentences and let $V$ denote the vocabulary size where the vocabulary is the unique set of sentences within a document pair. We follow previous works and use the Euclidean distance, $\Delta(i, j) = ||v_i - v_j||$ [148]. The XLSMD between a document pair is then the solution to the linear program:

$$XLSMD(A, B) = \min_{T \geq 0} \sum_{i=1}^{V} \sum_{j=1}^{V} T_{i,j} \times \Delta(i, j) \tag{6.3}$$

subject to:

$$\forall i \sum_{j=1}^{V} T_{i,j} = d_{A,i} \tag{6.4}$$

$$\forall j \sum_{i=1}^{V} T_{i,j} = d_{B,j} \tag{6.5}$$

Where $T \in R^{V \times V}$ is a nonnegative matrix, where each $T_{i,j}$ denotes how much of sentence $i$ in document $A$ is assigned to sentences $j$ in document $B$, and constraints ensure the flow of a given sentence cannot exceed its allocated mass. Specifically, XLSMD ensures the the entire outgoing flow from sentence $i$ equals $d_{A,i}$, i.e. $\sum_j T_{i,j} = d_{A,i}$. Additionally, the amount of incoming flow to sentence $j$ must match $d_{B,j}$, i.e., $\sum_i T_{i,j} = d_{B,j}$.

As described in Section 6.5, our competitive matching algorithm for aligning documents

relies on a similarity score. As such, before alignment, we transform each XLSMD into a similarity score as follows:

$$XLSMS(A, B) = e^{-XLSMD(A,B)} \tag{6.6}$$

Whereby two documents are more similar if the distance between them is smaller.

### 6.4.3 Alternative Sentence Weighting Schemes

In Equation 6.2, each document is represented as a normalized bag-of-sentences (nBOS). Under this assumptions, each sentence is considered equally important as a constituent of the document and the overall probability mass allocated to a sentence is proportional to the number of times it appears in a document. However, we posit that some sentences may be more semantically important than others within the same document and should therefore be allocated more mass. We investigate several weighting schemes to reflect these insights and evaluate their efficacy for document alignment in Section 6.7.2.

**Sentence Length Weighting** The first insight we investigate is that documents will naturally be segmented into sentences of different lengths based on the choice of sentence segmentation method, the language of the content int the document, and the content of a sentence. While Equation 6.2, treats each sentence equally, we posit that longer sentences should be assigned larger weighting than shorter sentences.

Under this weighting schema, each document is represented by a bag-of-sentences, but each sentence is weighted by the number of tokens in the sentence relative to the total number of tokens in the entire document, i.e., for the $i_{th}$ sentence in the document $A$,

$$d_{A,i} = \frac{count(i) \times |i|}{\sum_{s \in A} count(s) \times |s|} \tag{6.7}$$

where $|i|$ and $|s|$ indicate the number of tokens in sentence $i$ and sentence $s$ respectively. As such, longer sentence receive larger probability mass than shorter sentences. Once again, $d_{B,i}$ is computed in the same manner for document $B$.

**IDF Weighting** The second insight we investigate is that when mining for cross-lingual document pairs from a webdomain corpus, individual crawled documents contain many standard segments of text such as titles, column text, navigation text, etc. We believe that because this content is ubiquitous within the web-domain, it is less semantically informative

and should be allocated less weight when computing document distances. Based on this insight, we apply a variant of inverse document frequency (IDF) – a weighting scheme common in the information retrieval space – to individual sentences [158]. Under this scheme, the more common a sentence is within a webdomain, the less mass the sentence will be allocated.

We formalize IDF for a sentence $s$ in a webdomain-specific corpus $D$ as follows:

$$d_{A,i} = 1 + \log \frac{N+1}{1 + |\{d \in D : s \in d\}|} \tag{6.8}$$

where $N$ is the total number of web-documents in the web domain $D$, and $|\{d \in D : s \in d\}|$ is the number of documents where the sentence $s$ occurs. Smoothing by 1 is performed to prevent 0 IDF and division by zero.

As most sentences will occur only once within the web domain, they will have equal IDF weighting. Only *repetitive* sentences that are occur frequently within the web domain (e.g. boilerplate) will be down weighted.

**SLIDF Weighting** Finally, we propose combining both sentence length and inverse document frequency into a joint weighting scheme:

$$d_{A,i} = \frac{count(i) \times |i|}{\sum_{s \in A} count(s) \times |s|} \times \left(1 + \log \frac{N+1}{1 + |\{d \in D : s \in d\}|}\right) \tag{6.9}$$

In this scheme, each sentence is weighted proportionally to the number of tokens it contains as well as by the IDF of the sentence within the domain. This weighting scheme is reminiscent of the use of tf-idf to determine word relevance, but instead sentence length and idf are used to determine sentence importance [159].

### 6.4.4 Handling Imbalanced Document Mass

Many different aspects can lead to an unequal mass between source and target documents. One natural scheme considers that many document pairs contain an unequal number of sentences between the source document and target document. With an equal constant mass for each sentence, this naturally leads to unequal mass in the pair. In Section 6.4.3, alternate weighting schemes such as IDF and SLIDF can introduce unequal total mass between the source document and target document. As such, we must adapt earth mover's distance to handle computing a distance metric between documents of unequal mass.

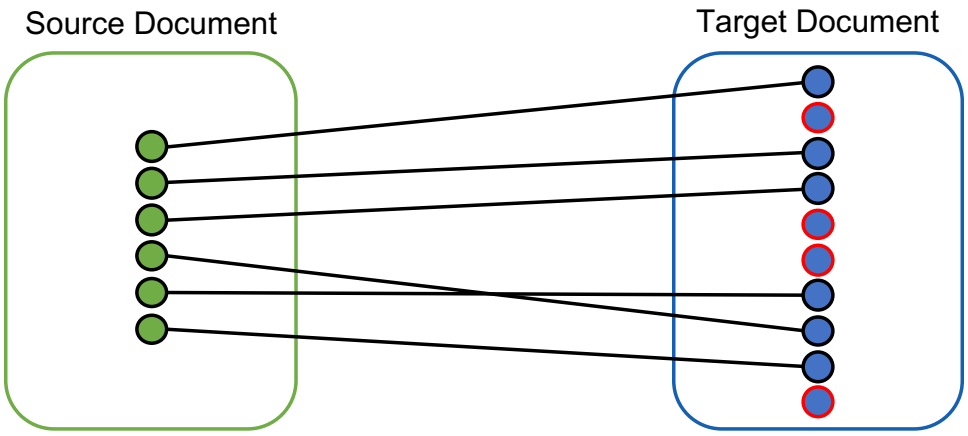



**Figure 6.3:** Assuming each sentence is associated with constant mass, this example portrays a document pair with a mis-matched number of sentences between the source and target documents and thus unequal mass between source and target documents. Sentences outlined in red signify left-over sentences after optimal alignment and mass transport between source and target sentences.

As seen in Figure 6.3, when computing the distance between two documents, the mass in the source document is not equal to the mass in the target document. Normally, EMD operates on a normalized histogram that induce a probability distribution with unit measure of 1. As a result, there is always equal mass in the source document and target document. However, this assumption doesn't hold in our investigation because we consider weighting schemes that may place a greater mass on one document than on another.

We propose three ways to address the imbalance between source and target documents that may occur due to the different weighting schemes we propose: (1) a no penalty evaluation (2) normalizing the mass in each document with any weighting scheme to unit measure 1 and (3) imposing an imbalance penalty for any left-over mass after optimal transportation calculation. In Section 6.6.1, we experimentally evaluate the impact these approaches have on downstream document alignment.

**No Penalty Evaluation** Under the no penalty evaluation, when the source and target documents have different mass, we allow for a partial matching between the source and target. That is, mass from sentences in the larger document is allocated to sentences in the smaller document. The left over (unmoved) mass from the larger document is then discarded *without penalty*. One caveat is that without a proper penalty, the imbalance causes this formulation to no longer be a true distance metric.

**Imbalance Penalty Evaluation** The second proposal allows for left over sentences to be destroyed from the larger of the two documents. However, there is a cost penalty for any leftover mass from sentences in the larger document. This cost penalty $\sigma$ signifies the penalty cost of one unit of leftover mass.

For the resulting distance to be a metric, $\sigma$ should be greater than or equal to half the diameter of the space i.e, the maximum possible distance between any two points. For our use case, we select the distance between the furthest two sentences between the source and target documents ensuring a proper resultant distance metric.

**Document Mass Normalization** The third option is to ensure that the two documents have the same mass regardless of the weighting scheme used. This can be done through normalizing the mass allocated to each sentence such that the total mass is of unit measure. We compute this normalization as follows:

$$d'_{A,i} = \frac{d_{A,i}}{\sum_{s \in A} d_{A,s}} \tag{6.10}$$

Consequently, by normalizing the mass to unit measure in both the source and target documents, each document has a legitimate distribution and the induced distance metric is valid.

### 6.4.5 Fast Distance Approximation

Previous works have shown that WMD achieves state-of-the-art results in many retrieval and classification tasks, WMD, and other EMD-based variants have been shown to suffer from high computational complexity $\mathcal{O}(p^3 \log p)$, where $p$ denotes the number of unique words in the each document pair.

**Relaxed XLSMD** Given the scalability challenges for computing WMD, simplified version of WMD was proposed that relaxes one of the two constraints in the original formulation [148]. Applying the same principle to XLSMD, we formulate:

$$XLSMD(A, B) = \min_{T \geq 0} \sum_{i=1}^{V} \sum_{j=1}^{V} T_{i,j} \times \Delta(i, j) \tag{6.11}$$

subject to: $\forall i \sum_{j=1}^{V} T_{i,j} = d_{A,i}$. Analogous to the relaxed-WMD, this relaxed problem yields a lower-bound to the XLSMD as every XLSMD solution satisfying both constraints

remains a feasible solution if one constraint is removed. The optimal solution to this relaxed formulation can be found by simply allocating the mass in each source sentence to the closest sentence in the target document as measured in the Euclidean embedding space.

The same computation can be performed in the reverse direction by removing the second constraint and keeping the first constraint: $\forall j \sum_{i=1}^{V} T_{i,j} = d_{B,j}$. In this scenario each sentence in the target document has its mass allocated to the closest sentence in the source document. Both these distances can be calculated by computing the distance matrix between all pairs of sentences in $\mathcal{O}(p^2)$ time. For a tighter estimate of distance, the maximum of the two resultant distances achieved from removing each of the constraints independently can be used.

**Greedy Mover's Distance** We introduce an alternative to the relaxed-EMD variant wherein we keep both constraints in the transportation problem, but identify an approximate transportation scheme, instead of solving for the optimal transport strategy. This proposed greedy approximation algorithm we dub "greedy mover's distance" (GMD) finds the two closest sentences and moves as much mass between the two sentences as possible; the algorithm moves to the next two closest pairs until all mass has been moved between the source and target document while maintaining both constraints.

**Algorithm 6.1:** Greedy Mover's Distance

**Input:** $d_s, d_t, w_s, w_t$
**Output:** $\Delta(d_s, d_t)$

```
pairs ← {(s_s, s_t) for s_s, s_t ∈ d_s × d_t}  in ascending order by  ||s_s − s_t||
distance ← 0.0
for s_s, s_t ∈ pairs do
    flow ← min(w_s[s_s], w_t[s_t])
    w_s[s_s] ← w_s[s_s] − flow
    w_t[s_t] ← w_t[s_t] − flow
    distance ← distance + ||s_s − s_t|| × flow
end
return total
```

As seen in Algorithm 6.1, the algorithm takes a source document ($d_s$) and a target document ($d_t$) as well as the weights for the sentences in each: respectively $w_s$ and $w_t$. The algorithm first computes the euclidean distance between each sentence pair from source to target and sorts these pairs in ascending order by their euclidean distance. The algorithm then iteratively chooses the closest sentence pair and moves the mass of the smallest between the two sentences. The remaining (unmoved) mass of each sentence is updated by

subtracting the moved mass from the unmoved mass. The total distance is updated by the amount of mass moved between the two sentences over the distance between the sentences. The algorithm terminates when all pairs of sentences have been observed and all moveable mass has been moved. Unlike the exact solution to EMD, the runtime complexity is a more tractable $\mathcal{O}(|d_s||d_t| \times \log(|d_s||d_t|))$ which is dominated by the cost of sorting all candidate pairs. Unlike the relaxation approximation, as both constraints must still hold, but the solution may not represent the optimal transport, this formulation yields an upper-bound to XLSMD. In the experiments section, we show that this approximation gives comparable distances to the exact EMD and the distances generated provide comparable downstream cross-lingual alignment results.

We experimentally compare the effect of both approximation strategies on downstream document alignment in Section 6.6.2.

## 6.5 DOCUMENT MATCHING ALGORITHM

In addition to a similarity metric (i.e. XLSMS), we need a document matching algorithm to determine the best mapping between documents in two languages. In our case, this works as follows: for any given webdomain, each document in the source document set, $D_s$ is paired with each document in the target set, $D_t$, yielding $|D_s \times D_t|$ scored pairs – a fully connected bipartite graph representing all candidate pairings. Similar to previous works, the expected output assumes that each webpage in the non-dominant language has a translated or comparable counterpart [138]. As visualized in Figure 6.1, this yields a $min(|D_s|, |D_t|)$ expected number of aligned pairs.

While an optimal matching maximizing scoring can be solved using the Hungarian algorithm [139], the complexity of this algorithm is $\mathcal{O}(max(|D_s||D_t|)^3)$ which is intractable to even moderately sized web domains. As such, similar to the work in [138], a one-to-one matching between English and non-English documents is enforced by applying, competitive matching, a greedy bipartite matching algorithm.

In Algorithm 6.2, the algorithm first scores each candidate document pair using the document similarity scoring function. These candidates are then sorted in order of most similar to least similar using their numerical score. The algorithm then iteratively chooses a document pair with the highest score as long as the $d_s$ and $d_t$ of each pair have not been used in a previous (higher scoring) pair. The algorithm terminates when $min(|D_s|, |D_t|)$ pairs have been selected. Unlike the Hungarian algorithm, the runtime complexity is a more tractable $\mathcal{O}(|D_s||D_t| \times \log(|D_s||D_t|))$ which is dominated by the cost of sorting all candidate pairs.

**Algorithm 6.2:** Competitive Matching

**Input:** $P = \{(d_s, d_t) | d_s \in D_s, d_t \in D_t\}$
**Output:** $P' = \{(d_{s,i}, d_{t,i}), ...\} \subset P$

```
scored ← {(p, score(p)) for p ∈ P}
sorted ← sort(scored) in descending order
aligned ← ∅
S_s ← ∅
S_t ← ∅
for d_s, d_t ∈ sorted do
    if d_s ∉ S_s ∧ d_t ∉ S_t then
        aligned ← aligned ∪ {(d_s, d_t)}
        S_s ← S_s ∪ d_s
        S_t ← S_t ∪ d_t
end
return aligned
```

## 6.6 ANALYSIS

In this section, we analyze the performance of XLSMD under additional conditions. First we investigate the effects of different approaches to account for imbalanced document sizes. Second, we explore the effect of choosing faster approximation algorithms to speed-up distance computation.

### 6.6.1 Document Imbalance Experiments

While most implementations of EMD measure the distance between two distributions, in Section 6.4.3, we introduce several weighting schemes that do not constitute probability distributions. In Section 6.4.4, we note that this can lead to document imbalance whereby the source and target documents have unequal total mass and proposed three approaches to addressing unequal mass.

We pick a variant of XLSMD (SLIDF) perform document alignment on a selection of low, mid, and high-resource directions. For each direction, we evaluate the distance with three approaches to handling document mass imbalance (1) no penalty (2) max distance penalty and (3) normalizing weights.

In Table 6.1 we report the average document alignment recall for low, mid, and high-resource language pairs for each technique for handling document imbalance. We observe that the worst-performing technique is to calculate the distance without imposing some penalty when imbalance is present. We posit this is because scenarios can arise where a document with a small amount of content can be paired with a document with a large amount

| Approach | Low | Mid | High | All |
|---|---|---|---|---|
| No Penalty | 0.44 | 0.37 | 0.44 | 0.40 |
| Penalty | 0.47 | 0.44 | 0.50 | 0.46 |
| Normalization | 0.55 | 0.52 | 0.57 | 0.54 |

**Table 6.1:** Evaluating approaches for handling document-mass imbalance due to alternative sentence weighting.

of content due to the smaller content having sentences semantically close to many sentences in the larger document. However, such pairings are not necessarily good pairs. Imposing a penalty appears to mitigate this and outperforms no penalty across low, mid, and high-resource language pairs. However, consistently normalizing the unbalanced documents each to unit measure as specified in Equation 6.10 consistently outperforms both the no-penalty and penalty approaches to handling imbalance.

### 6.6.2 Distance Computation Experiments

Although using sentences over words as the base semantic unit drastically reduces the overall cost of computing EMD-based metrics, the cubic computation still prohibits its use as a fast similarity metric for large-scale alignment efforts. As such, in Section 6.4.5 we described two approximations to EMD computation: (1) a relaxation of constraints and (2) a greedy algorithm for computing EMD. Using these two techniques, we can significantly speed up the distance computations between document pairs. However, the constraint relaxation and greedy algorithm for computing distances represent a lower-bound and upper bound respectively on the true XLSMD.

We first analyze and compare the distances from each approximation scheme to the true XLSMD.

| Method | Kendall-Tau | Recall | MAE | Runtime (s) |
|---|---|---|---|---|
| Exact-XLSMD | 1.00 | 0.69 | 0.000 | 0.402 |
| Relaxed-XLSMD | 0.70 | 0.58 | 0.084 | 0.031 |
| Greedy-XLSMD | 0.98 | 0.69 | 0.010 | 0.107 |

**Table 6.2:** Comparing exact XLSMD computation to approximation schemes for computing XLSMD on 10 webdomains.

In Figure 6.7, we see that the distance computations for exact XLSMD and the greedy XLSMD approximation are highly correlated with small variance, while the relaxed approximation is less so with high variance. Additionally, as discussed in Section 6.4.5, the visualizations verify that our greedy approximation is a fairly tight upper bound while the relaxed approximation is a looser lower bound.

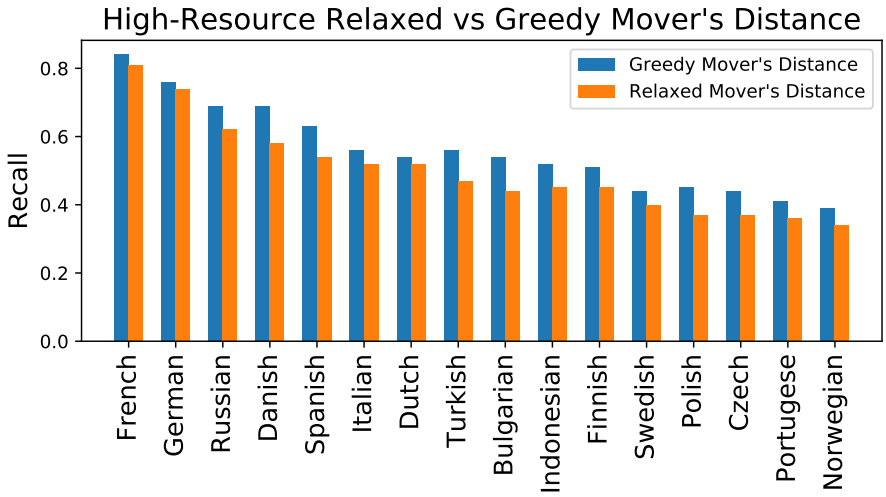



**Figure 6.4:** Recall for distance approximation schemes for high-resource directions.

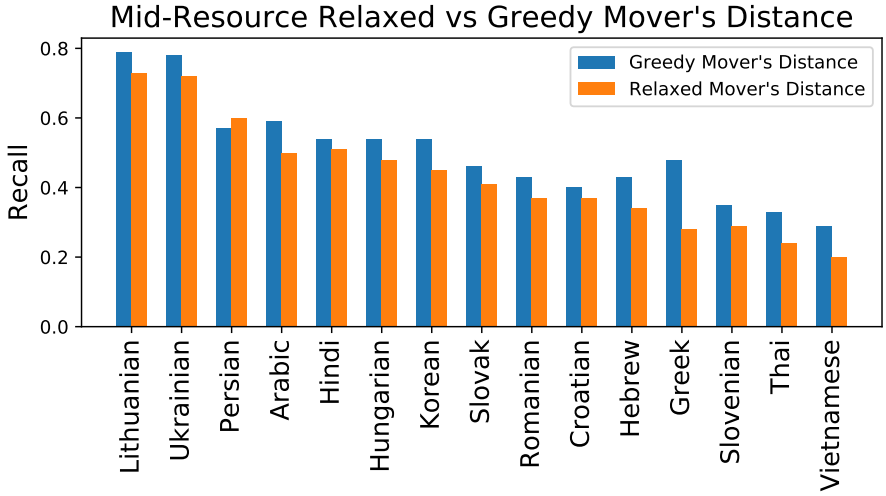



**Figure 6.5:** Recall for distance approximation schemes for mid-resource directions.

In Table 6.2, we compare quantitative metrics for the relaxed and greedy approximations to the exact solution of XLSMD on ten webdomains. Our first evaluation investigates how the approximate computation of distances affects the ordering or document pairs. For the ten selected webdomains, we sort the document pairs in order by their computed distances and compare the ordering to the ordering induced by the exact computation of XLSMD. We evaluate the orderings using the Kendall-Tau metric [160]. This correlation coefficient measures the agreement between the two rankings; if the agreement between the two rankings is perfect (i.e., the two rankings are the same) the coefficient has value 1 and if the disagreement between the two rankings is perfect (i.e., one ranking is the reverse of the other) the coefficient has value -1. Intuitively, we would like the distances computed by an

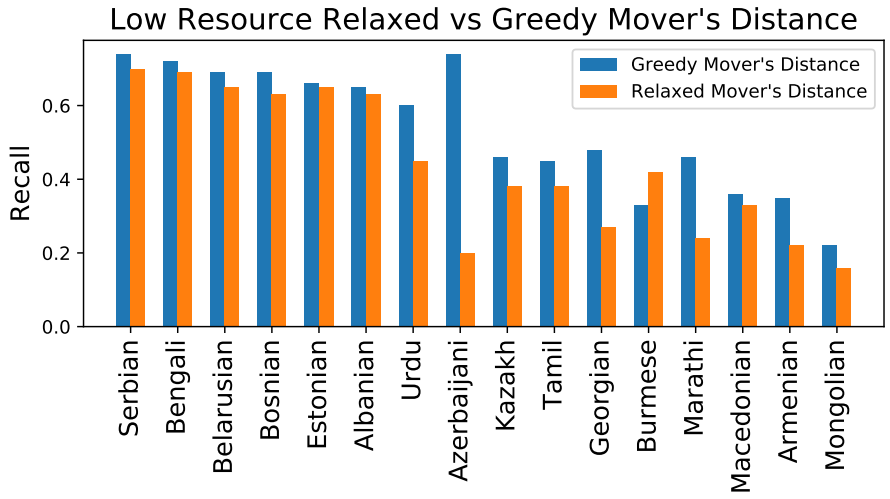



**Figure 6.6:** Recall for distance approximation schemes for low-resource directions.

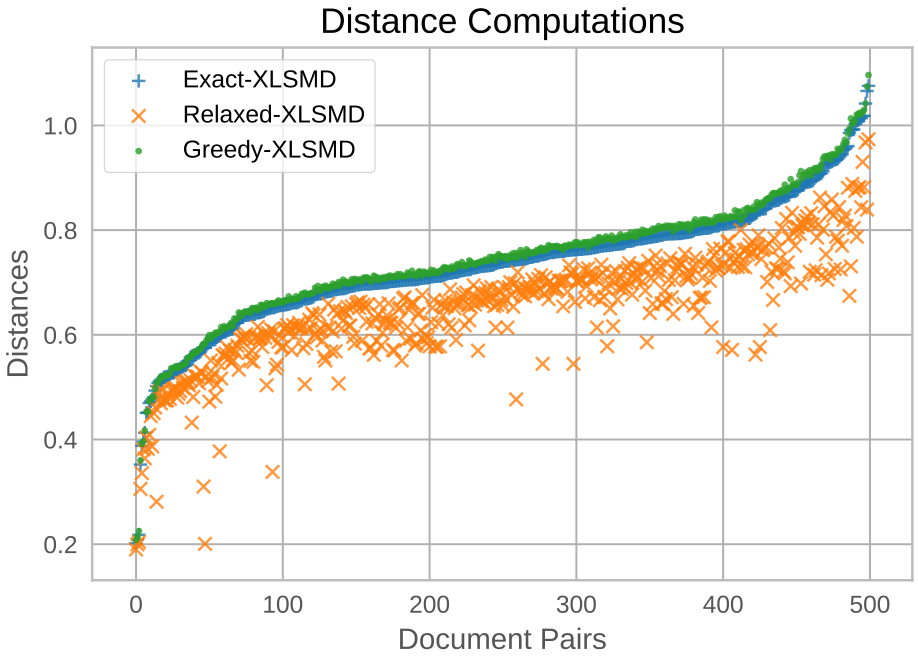



**Figure 6.7:** Exact, relaxed, and greedy-XLSMD distances sorted by Exact-XLSMD for a random selection of document pairs.

approximation to induce a similar ordering to the ordering by the exact distance computation. Comparing the Kendall-Tau for the relaxed and greedy approximations in relation to the exact computation shows that the order induced by the greedy approximation is very similar to the ordering induced by the exact computation while the relaxed approximation varies considerably. Additionally, the relaxed approximation demonstrates fairly high mean absolute error (MAE) and results in lower document alignment recall when compared to

the exact computation of XLSMD, while our greedy approximation performs comparably and shows insignificant MAE. Finally, while the runtime of the relaxed computation is the fastest at 13 times faster than the exact computation, our greedy algorithm is approximately 4 times faster while delivering comparable document alignment performance to the exact computation and superior performance to the relaxed computation.

To ensure that the greedy algorithm consistently outperforms the relaxed algorithm on document alignment, we investigate the effect of using each approximation method on the downstream document alignment performance across 47 language pairs of varying resource availability. We do not report results from the exact XLSMD distance as it was not tractable to run on the 47 evaluated language pairs.

| **Approximation** | **Low** | **Mid** | **High** | **All** |
|---|---|---|---|---|
| Relaxed-XLSMD | 0.44 | 0.43 | 0.50 | 0.46 |
| Greedy-XLSMD | 0.54 | 0.50 | 0.56 | 0.54 |

**Table 6.3:** Document alignment performance of fast methods for approximating the same variant of XLSMD.

As seen in Figures 6.4,6.5, and 6.6, in 45 of the 47 evaluated language pairs, our proposed Greedy Mover's Distance approximation yielded higher downstream recall in our alignment task over using the relaxed distance proposed for use in WMD [148]. In Tables 6.3, we see a 10%, 7%, and 6% improvement in downstream recall across low, mid, and high-resource directions respectively. These results indicate that relaxing one of the two constraints in EMD is too lax for measuring an accurate distance. We posit this is because there are many sentences that can be considered "hubs" that are semantically close to many other sentences. These sentences can have a lot of probability mass allocated to them, resulting in a lower approximate EMD. Our greedy approximation ensures that both constraints are maintained even if the final result is not the minimum distance between the two.

## 6.7 EXPERIMENTS AND RESULTS

In this section, we explore the question of whether XLSMS can be used as a similarity metric for the document alignment problem. Moreover, we explore what are the different variants of weightings that yield the best results.

### 6.7.1 Experimental Setup

**Dataset** We evaluate on the test set from the URL-Aligned CommonCrawl dataset [6]. This dataset consists of a massive collection of 54 million web documents in non-English

| | Recall | | | | | |
|---|---|---|---|---|---|---|
| **Language** | **DirectEmb** | **SentAvg** | **XLSMD** | **+SL** | **+IDF** | **+SLIDF** |
| French | 0.39 | 0.84 | 0.81 | 0.84 | 0.83 | **0.85** |
| Spanish | 0.34 | 0.53 | 0.59 | 0.63 | 0.62 | **0.64** |
| Russian | 0.06 | 0.64 | 0.69 | 0.69 | 0.70 | **0.71** |
| German | 0.52 | 0.74 | **0.78** | 0.76 | 0.77 | 0.77 |
| Italian | 0.22 | 0.47 | 0.55 | 0.56 | 0.56 | **0.59** |
| Portuguese | 0.17 | 0.36 | 0.39 | **0.41** | 0.38 | 0.40 |
| Dutch | 0.28 | 0.49 | 0.54 | 0.54 | 0.54 | **0.56** |
| Indonesian | 0.11 | 0.47 | 0.49 | 0.52 | 0.51 | **0.53** |
| Polish | 0.17 | 0.38 | 0.45 | 0.45 | **0.46** | **0.46** |
| Turkish | 0.12 | 0.38 | 0.52 | 0.56 | 0.57 | **0.59** |
| Swedish | 0.19 | 0.40 | 0.44 | 0.44 | **0.46** | 0.45 |
| Danish | 0.27 | 0.62 | 0.63 | **0.69** | 0.65 | **0.69** |
| Czech | 0.15 | 0.40 | 0.43 | **0.44** | **0.44** | 0.43 |
| Bulgarian | 0.07 | 0.43 | 0.52 | 0.54 | **0.55** | 0.52 |
| Finnish | 0.06 | 0.47 | 0.51 | 0.51 | **0.54** | 0.52 |
| Norwegian | 0.13 | 0.33 | 0.37 | 0.39 | **0.42** | 0.41 |
| **AVG** | 0.20 | 0.50 | 0.54 | 0.56 | 0.56 | **0.57** |

**Table 6.4:** Recall on high-resource language directions.

languages aligned with their English translation. The document pairs cover 92 language directions covering languages varying in resource availability, language family, and morphology. 47 language directions across low, mid, and high resource directions were selected for evaluation.

**Baseline Methods** For comparison, we implemented two existing and intuitive document scoring baselines previously evaluated on this URL-Aligned CommonCrawl dataset [6]. The first method dubbed direct embedding (DirectEmb) treats the entire content of a document as a single input and embeds the document into a multilingual space using LASER; documents are then compared by computing cosine similarity between document representations. The second baseline performs document embedding by segmenting each document into smaller sentences, performing embedding at the sentence level, then averaging all sentence embeddings to form a document representation; once again documents are compared by computing cosine similarity between their dense representations. For consistency, all multilingual representations used for this experiment were performed using LASER embeddings.

| | Recall | | | | | |
|---|---|---|---|---|---|---|
| **Language** | **DirectEmb** | **SentAvg** | **XLSMD** | **+SL** | **+IDF** | **+SLIDF** |
| Romanian | 0.15 | 0.40 | 0.44 | 0.43 | **0.45** | 0.43 |
| Vietnamese | 0.06 | 0.28 | 0.29 | 0.29 | 0.29 | **0.32** |
| Ukrainian | 0.05 | 0.68 | 0.67 | 0.78 | 0.78 | **0.82** |
| Greek | 0.05 | 0.31 | 0.47 | 0.48 | **0.49** | **0.49** |
| Korean | 0.06 | 0.34 | 0.60 | 0.54 | **0.61** | 0.60 |
| Arabic | 0.04 | 0.32 | 0.63 | 0.59 | **0.65** | 0.61 |
| Croatian | 0.16 | 0.37 | 0.40 | 0.40 | **0.41** | 0.40 |
| Slovak | 0.20 | 0.41 | 0.46 | **0.46** | **0.46** | 0.44 |
| Thai | 0.02 | 0.19 | 0.41 | 0.33 | **0.47** | 0.41 |
| Hebrew | 0.05 | 0.18 | 0.39 | **0.43** | 0.41 | 0.41 |
| Hindi | 0.04 | 0.27 | 0.34 | **0.54** | 0.52 | 0.53 |
| Hungarian | 0.15 | 0.49 | 0.50 | **0.54** | 0.51 | **0.54** |
| Lithuanian | 0.11 | 0.73 | 0.79 | 0.79 | **0.80** | **0.80** |
| Slovenian | 0.13 | 0.33 | 0.34 | 0.35 | **0.36** | **0.36** |
| Persian | 0.06 | 0.32 | 0.56 | 0.57 | 0.53 | **0.59** |
| **AVG** | 0.09 | 0.37 | 0.49 | 0.50 | **0.52** | **0.52** |

**Table 6.5:** Recall on mid-resource language directions.

**XLSMD Weightings** We investigate variants of our XLSMD using four different weighting schemes: (1) vanilla XLSMD with each sentence equally weighted within each document (2) weighting by sentence length (+SL) where XLSMD is computed under a scheme where each sentence is weighted by its length (number of tokens) normalized by the length of the entire document (3) weighting by inverse document frequence (+IDF) where XLSMD is computed under a scheme where each sentence is weighted by the idf of the sentence (4) computing XLSMD under a scheme where each sentence is weighted by both sentence length and inverse document frequency (+SLIDF).

**Normalization** For our experiments, we use document mass normalization to deal with imbalanced document mass. In Section 6.6.1 we present ablation results on different techniques for handling unbalanced document mass.

**Distance approximation** We use the greedy mover's distance approximation for all variants reported. In Section 6.6.2 we further explore the performance of the full distance computation and relaxed variants that were described in Section 6.4.5.

| | Recall | | | | | |
|---|---|---|---|---|---|---|
| **Language** | **DirectEmb** | **SentAvg** | **XLSMD** | **+SL** | **+IDF** | **+SLIDF** |
| Estonian | 0.28 | 0.52 | 0.69 | 0.66 | **0.74** | 0.72 |
| Bengali | 0.05 | 0.32 | 0.78 | 0.72 | 0.77 | **0.79** |
| Albanian | 0.23 | 0.56 | **0.66** | 0.65 | 0.65 | **0.66** |
| Macedonian | 0.02 | 0.33 | 0.32 | 0.36 | **0.38** | 0.33 |
| Urdu | 0.06 | 0.22 | **0.60** | **0.60** | 0.49 | 0.56 |
| Serbian | 0.06 | 0.59 | **0.75** | 0.74 | 0.74 | 0.71 |
| Azerbaijani | 0.08 | 0.34 | 0.74 | 0.74 | **0.75** | 0.74 |
| Armenian | 0.02 | 0.18 | 0.32 | 0.35 | 0.34 | **0.38** |
| Belarusian | 0.07 | 0.47 | 0.67 | 0.69 | **0.73** | 0.71 |
| Georgian | 0.06 | 0.24 | 0.46 | **0.48** | 0.45 | 0.45 |
| Tamil | 0.02 | 0.20 | 0.51 | 0.45 | 0.51 | **0.53** |
| Marathi | 0.02 | 0.11 | 0.43 | **0.46** | 0.33 | 0.39 |
| Kazakh | 0.05 | 0.31 | 0.44 | **0.46** | 0.45 | 0.45 |
| Mongolian | 0.03 | 0.13 | 0.18 | 0.22 | 0.21 | **0.23** |
| Burmese | 0.01 | 0.10 | 0.26 | 0.33 | **0.46** | **0.46** |
| Bosnian | 0.18 | 0.64 | 0.61 | 0.69 | 0.65 | **0.72** |
| **AVG** | 0.08 | 0.33 | 0.53 | 0.54 | 0.54 | **0.55** |

**Table 6.6:** Recall on low-resource language directions.

**Evaluation Metric for Document Alignment** Because the ground-truth document pairs only reflect a high-precision set of web-document pairs that are translations or of comparable content, there may be many other valid cross-lingual document pairs within each web-domain that are not included in the ground truth set. As such, we evaluate each method's generated document pairs solely on the recall (i.e. what percentage of the aligned pages in the test set are found) from the ground truth pairs.

For each scoring method, we score document pairs from the source and target languages within the same webdomain using the proposed document similarity metrics described above. For the alignment, we report the performance for each document similarity measure after applying the competitive matching alignment algorithm as described in Algorithm 5.1. Applying 1-to-1 matching has been shown to not only improving the resultant alignment pairs, but also ensures the each method produces the same number of aligned pairs to allow for a fair comparison of recall scores.

### 6.7.2 Results

In Tables 6.4,6.5, and 6.6, we compare our proposed method to various baselines in identifying ground-truth aligned document pairs. We first notice that constructing document representations by directly embedding (DirectEmb) the entire content of each document and computing document similarity using cosine similarity of the representation severely underperforms compared to individually embedding sentences and constructing the document representations by averaging the individual sentence representations within the document (SentAvg). This is intuitive as LASER embeddings were trained on parallel sentences and embedding larger documents directly using LASER results in poorer representations than by first embedding smaller sentences and combining them into the final document representation.

Comparing the basic XLSMD to the best performing baseline (SentAvg), we see a 4%, 12%, and 20% improvement across high, mid, and low-resource directions respectively. This improvement suggests that summing sentence embeddings into a single document representation degrades the quality of the resultant document distances over computing document distances by keeping all sentence representations separate and computing distances between individual sentence pairs and combining these distances into a final document distance. This is more pronounced in lower-resource over higher-resource pairs which we theorize is due to the quality of lower-resource embeddings being worse due LASER being trained on fewer low-resource sentence pairs. As such averaging is more destructive to these representations while XLSMD avoids this degradation.

Further analyzing the results by comparing the four variants we proposed for XLSMD, we verify our intuitions that different sentences should be allocated different weighting when computing document distances. When we assign mass to each sentence proportional the number of tokens in the sentence (+SL), we see a 2%, 1% and 1% absolute improvement in recall in high, mid, and low-resource directions over assigning equal probability mass to each sentence. This supports our claim that as segmenting documents yields a bag-of-sentences representation whereby sentences are of different sizes, we should allocate more importance to longer sentences over shorter sentences as they contain more semantic content. The second assumption we investigated is that sentences that are common within a webdomain have less semantic importance and should be allocated less probability mass when computing XLSMD. After computing XLSMD with each sentence allocated mass according to inverse document frequency (+IDF) and normalized to unit measure, we see a 2%, 3%, and 1% improvement over the baseline equal weighting among sentences. This verifies our assumption that sentences that are common within a webdomain are likely boilerplate (col-

umn names, navigation buttons, recurring titles, etc) and less important when measuring semantic distance between document content. Finally, we investigate the performance of document alignment after combining both sentence length and inverse document frequency weighting to assign probability mass to each sentence (+SLIDF). Falling in line with our intuition, we see a 3%, 3% and 2% absolute improvement in recall for high, mid, and low-resource directions respectively over the the approach that equally weights each sentence. Overall, our XLSMD with SLIDF weighting scheme to assigning probability mass to sentences outperforms the sentence averaging baseline by 7% on high-resource directions, 15% on mid-resource directions, and 22% on low-resource directions.

## 6.8 DISCUSSION

In this chapter, we introduce XLSMD a cross-lingual sentence mover's distance metric for automatically assessing the semantic similarity of two documents in different languages. We leverage state-of-the-art multilingual sentence embeddings and apply XLSMD to the task of cross-lingual document alignment. We demonstrate that our new metric outperforms other unsupervised metrics by a margin, especially in medium and low-resourced conditions.

Recognizing that solving for the exact solution of XLSMD becomes computationally intractable for long web-documents and large-scale document alignment, we introduce a fast approximation scheme with comparable performance to exact computation.

One natural extension of this work is to further investigate weighting schemes. As seen in our results, choosing a proper weighting scheme can significantly improve the performance of downstream document alignment. A natural extension from this unsupervised process is a supervised model that might better guide the cross-lingual alignment process than the unsupervised distance metric used here. Several approaches could be investigated to incorporate supervision including metric learning and directly learning document representations to discern cross-lingual documents.

Another area of investigation is in better cross-lingual representations. Currently, the cross-lingual representations leverage subwords via byte pair encoding and a sequence-to-sequence model to learn sentence representations. Improvements would leverage all the granularity levels mentioned in previous chapters. Ideally in addition to subwords, the learned representations should leverage not only subwords, but also words and phrases hierarchically to learn the cross-lingual representations.

# CHAPTER 7: SUMMARY & DISCUSSION

While many techniques utilize word tokens as the base semantic units, there are many choices at different semantic granularity that can be utilized by natural language processing techniques. In this work, we demonstrate how using subword-level, phrase-level, and sentence-level granularity in addition to standard token-level inputs can improve a variety of downstream text-mining tasks. We argue that utilizing granularity at multiple levels is necessary to capture the correct semantic meaning depending on the task.

We posit that these three segmentations into subwords, phrases, and sentences are fundamental techniques that should be performed as as the first step to many text-mining tasks. As such, we not only introduce techniques for segmenting text into the proper granularity, but also investigate models leverage input at multiple granularity to improve performance over base word segmentations.

1. **Subword mining with an application to word embeddings:** We developed unsupervised and supervised subword segmentation algorithms and demonstrated that incorporating these subwords along with the original word enriches word embeddings, resulting in improved embeddings and superior performance in downstream embedding and language modeling tasks.

2. **Phrase mining with an application to topic modeling:** We developed an unsupervised phrasal segmentation algorithm and applied the segmentation to a downstream topic modeling task. We demonstrated experimentally that our method is scalable, provides high-quality phrases, and results in interpretable topics.

3. **Cross-lingual sentence representations with an application to mining parallel data:** We motivated the use of sentence-level cross-lingual representations as a tool for identifying parallel text. We describe techniques for identifying parallel documents in many languages as well as techniques for mining parallel sentences from these aligned documents. We demonstrate experimentally that this mined data can be a valuable source of training data for machine translation.

## 7.1 HIERARCHICAL RELATIONSHIPS BETWEEN SEMANTIC UNITS

One common motif throughout the works is the hierarchical nature of semantic units. This notion is that each granularity represents a level of semantic information. The finer the granularity the more fundamental and simple of a concept is encoded in the semantic unit.

The coarser the granularity the more high-level and complex an idea is represented within the unit.

While we are more accustomed to individual words as a base semantic units, in Chapters 2 and 3, we demonstrated that these words could be decomposed into semantically meaningful subword structures. From another perspective, words in morphologically rich languages can be constructed from base morphemes and subwords and by combining several morphemes, a more informative semantic word unit is formed.

Similarly in Chapter 4, we demonstrated that individual words at time may not present the correct level of semantic meaning, and at times combining several words together captures the intended semantic intent. This has been independently identified in previous works whereby search and information retrieval benefit from grouping frequent, less informative unigrams into ngrams for retrieval [1]. Other works have noted that performing information extraction on unstructured text data involves grouping individual words into phrases that correspond into entities and relations [161, 162, 163].

Finally in Chapters 5 and 6, we investigate the use of sentence-level semantic units. These sentence level units were constructed from decomposing sentences into words and subwords, and learning cross-lingual representations through a sequence-to-sequence task. While subwords and words were leveraged in these representations to mitigate data sparsity, it is an open question as to the benefit of leveraging phrase-level input as well which could capture higher-order semantic concepts. Additionally, in Chapter 5, we demonstrate the benefit of combining sentence-level units to form meaningful document representations.

As shown in these studies, each level of semantic granularity can be composed from smaller semantic units. By combining base-units into coarse units, more complex concepts and information can be encoded. Simultaneously including finer-grained units can provide simpler semantic information whose meanings can supplement these high-level units. This supports the claim that leveraging text input at multiple granularity provides a more holistic representation that can be more fully leveraged to improve modeling and understanding for downstream tasks.

## 7.2 CONCLUSIONS

In summary, this dissertation provides evidence to support the following statement: identifying and leveraging multiple levels of semantic granularity is crucial to text mining and natural language processing. To this end, we introduce several techniques to decompose words into finer-grained semantic units, and combine words and subwords into coarser-grained semantic units. We analyze several text mining tasks including learning distributed

word representations, topic modeling, and mining parallel data and demonstrate how incorporating text at different levels of semantic granularity improve each of these tasks.

## 7.3 FUTURE WORKS

While the works discussed give a preliminary view of leveraging semantic input units at multiple granularity for more effective text understanding, there are many further works and unexplored areas to investigate.

One area open to further investigation of how the hierarchy of semantic units at different granularity interact. Different natural language processing tasks may be better suited to certain levels of semantic granularity over others. What determines the suitability of a granularity for a particular task is an important aspect to understand. Additionally, further investigation into how semantic units can be combined to coarser-grained units can aid in development of new representations such as better word embeddings, sentence embeddings, paragraph embeddings, and document embeddings.

Another area of future work includes the development of metrics to assess the suitability of certain languages to decomposition into certain levels of granularity. For example, many languages are morphologically rich, others are agglutinative, while others may lack morphological richness. As such, not all languages may be predisposed to subword segmentation. Automatically developing metrics to assess the predisposition of a language to certain levels of granularity can aid in deciding the proper level of segmentation to utilize.

## REFERENCES


[1] G. Salton and C. T. Yu, "On the construction of effective vocabularies for information retrieval," *Acm Sigplan Notices*, vol. 10, no. 1, pp. 48–60, 1973.

[2] A. Kornai, *Mathematical linguistics*. Springer Science & Business Media, 2007.

[3] A. El-Kishky, F. Xu, A. Zhang, and J. Han, "Parsimonious morpheme segmentation with an application to enriching word embeddings," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 64–73.

[4] A. El-Kishky, X. Fu, A. Addawood, N. Sobh, C. Voss, and J. Han, "Constrained sequence-to-sequence semitic root extraction for enriching word embeddings," in *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, 2019, pp. 88–96.

[5] A. El-Kishky, Y. Song, C. Wang, C. R. Voss, and J. Han, "Scalable topical phrase mining from text corpora," *Proceedings of the VLDB Endowment*, vol. 8, no. 3, 2014.

[6] A. El-Kishky, V. Chaudhary, F. Guzman, and P. Koehn, "A massive collection of cross-lingual web-document pairs," *arXiv preprint arXiv:1911.06154*, 2019.

[7] A. El-Kishky and F. Guzmán, "Massively multilingual document alignment with cross-lingual sentence-mover's distance," *arXiv preprint arXiv:2002.00761*, 2020.

[8] H. Hammarström and L. Borin, "Unsupervised learning of morphology," *Computational Linguistics*, vol. 37, no. 2, pp. 309–350, 2011.

[9] M. Creutz and K. Lagus, "Unsupervised discovery of morphemes," in *Proceedings of the ACL-02 workshop on Morphological and phonological learning-Volume 6*. Association for Computational Linguistics, 2002, pp. 21–30.

[10] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *Proc. ACL*, 2016.

[11] T. Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 66–75.

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[13] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[14] T. Mikolov, W.-t. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations." in *hlt-Naacl*, vol. 13, 2013, pp. 746–751.

[15] A. Joulin, E. Grave, and P. B. T. Mikolov, "Bag of tricks for efficient text classification," *EACL 2017*, p. 427, 2017.

[16] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[17] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning sentiment-specific word embedding for twitter sentiment classification." in *ACL (1)*, 2014, pp. 1555–1565.

[18] W. Y. Zou, R. Socher, D. Cer, and C. D. Manning, "Bilingual word embeddings for phrase-based machine translation," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 1393–1398.

[19] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 160–167.

[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[21] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[22] A. El-Kishky, F. Xu, A. Zhang, S. Macke, and J. Han, "Entropy-based subword mining with an application to word embeddings," in *SCLeM*, 2018, pp. 12–21.

[23] Z. S. Harris, "From phoneme to morpheme," in *Papers in Structural and Transformational Linguistics*. Springer, 1970, pp. 32–67.

[24] M. A. Hafer and S. F. Weiss, "Word segmentation by letter successor varieties," *Information storage and retrieval*, vol. 10, no. 11-12, pp. 371–385, 1974.

[25] H. Déjean, "Morphemes as necessary concept for structures discovery from untagged corpora," in *Proceedings of the Joint Conferences on New Methods in Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, 1998, pp. 295–298.

[26] J. R. Saffran, E. L. Newport, and R. N. Aslin, "Word segmentation: The role of distributional cues," *Journal of memory and language*, vol. 35, no. 4, pp. 606–621, 1996.

[27] S. Neuvel and S. A. Fulop, "Unsupervised learning of morphology without morphemes," in *Proceedings of the ACL-02 workshop on Morphological and phonological learning-Volume 6*. Association for Computational Linguistics, 2002, pp. 31–40.

[28] P. Schone and D. Jurafsky, "Is knowledge-free induction of multiword unit dictionary headwords a solved problem," in *Proc. EMNLP*, 2001, pp. 100–108.

[29] H. Sak, M. Saraclar, and T. Güngör, "Morphology-based and sub-word language modeling for turkish speech recognition," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. IEEE, 2010, pp. 5402–5405.

[30] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, and K. Macherey, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[31] M. Schuster and K. Nakajima, "Japanese and korean voice search," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 5149–5152.

[32] A. Alexandrescu and K. Kirchhoff, "Factored neural language models," in *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*. Association for Computational Linguistics, 2006, pp. 1–4.

[33] Q. Cui, B. Gao, J. Bian, S. Qiu, H. Dai, and T.-Y. Liu, "Knet: A general framework for learning word embedding using morphological knowledge," *ACM Transactions on Information Systems (TOIS)*, vol. 34, no. 1, p. 4, 2015.

[34] S. Qiu, Q. Cui, J. Bian, B. Gao, and T.-Y. Liu, "Co-learning of word representations and morpheme representations." in *COLING*, 2014, pp. 141–150.

[35] R. Cotterell and H. Schütze, "Morphological word-embeddings." in *HLT-NAACL*, 2015, pp. 1287–1292.

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

[37] P. Bojanowski, A. Joulin, and T. Mikolov, "Alternative structures for character-level rnns," *Prof. ICLR*, 2015.

[38] I. Sutskever, J. Martens, and G. E. Hinton, "Generating text with recurrent neural networks," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 1017–1024.

[39] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-aware neural language models." in *AAAI*, 2016, pp. 2741–2749.

[40] H. G. Gauch, *Scientific method in practice*. Cambridge University Press, 2003.

[41] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

[42] M. Kurimo, S. Virpioja, V. Turunen, and K. Lagus, "Morpho challenge competition 2005–2010: evaluations and results," in *ACL Special Interest Group on Computational Morphology and Phonology*. ACL, 2010, pp. 87–95.

[43] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin, "Placing search in context: The concept revisited," *ACM Transactions on information systems*, vol. 20, no. 1, pp. 116–131, 2002.

[44] F. Hill, R. Reichart, and A. Korhonen, "Simlex-999: Evaluating semantic models with (genuine) similarity estimation," *Computational Linguistics*, vol. 41, no. 4, pp. 665–695, 2015.

[45] T. Luong, R. Socher, and C. Manning, "Better word representations with recursive neural networks for morphology," in *CoNLL*, 2013, pp. 104–113.

[46] S. Barzegar, B. Davis, M. Zarrouk, S. Handschuh, and A. Freitas, "Semr-11: A multilingual gold-standard for semantic similarity and relatedness for eleven languages," in *LREC*, 2018.

[47] G. Ercan and O. T. Yıldız, "Anlamver: Semantic model evaluation dataset for turkish-word similarity and relatedness," in *COLING*, 2018, pp. 3819–3836.

[48] M. Köper, C. Scheible, and S. S. im Walde, "Multilingual reliability and" semantic" structure of continuous word spaces," in *IWCS*, 2015, pp. 40–45.

[49] G. Sahin, "Classification of turkish semantic relation pairs using different sources," *International Journal of Computer Engineering and Information Technology*, vol. 8, no. 10, p. 196, 2016.

[50] M. Faruqui, Y. Tsvetkov, P. Rastogi, and C. Dyer, "Problems with evaluation of word embeddings using word similarity tasks," in *RepEval*, 2016, pp. 30–35.

[51] L. Egghe, "Untangling herdan's law and heaps' law: Mathematical and informetric arguments," *Journal of the American Society for Information Science and Technology*, vol. 58, no. 5, pp. 702–709, 2007.

[52] K. Darwish, "Building a shallow arabic morphological analyzer in one day," in *ACL-02 workshop on Computational approaches to semitic languages*, 2002.

[53] E. Daya, D. Roth, and S. Wintner, "Identifying semitic roots: Machine learning with linguistic constraints," *Computational Linguistics*, 2008.

[54] N. Y. Habash, "Introduction to arabic natural language processing," *Synthesis Lectures on Human Language Technologies*, 2010.

[55] K. Taghva, R. Elkhoury, and J. Coombs, "Arabic stemming without a root dictionary," in *Information Technology: Coding and Computing*, 2005.

[56] S. Khoja and R. Garside, "Stemming arabic text," *Lancaster, UK, Computing Department, Lancaster University*, 1999.

[57] T. Zerrouki, "Tashaphyne, arabic light stemmer/segment," 2010.

[58] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[59] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[60] L. S. Larkey, L. Ballesteros, and M. E. Connell, "Improving stemming for arabic information retrieval: light stemming and co-occurrence analysis," in *ACM SIGIR*, 2002.

[61] M. Aljlayl and O. Frieder, "On arabic search: improving the retrieval effectiveness via a light stemming approach," in *CIKM*, 2002.

[62] M. Momani and J. Faraj, "A novel algorithm to extract tri-literal arabic roots," in *Computer Systems and Applications, 2007. AICCSA'07.*, 2007.

[63] Y. K. Lee, A. Haghighi, and R. Barzilay, "Modeling syntactic context improves morphological segmentation," in *CoNLL*, 2011.

[64] A. Boudlal, M. O. A. O. Bebah, A. Lakhouaja, A. Mazroui, and A. Meziane, "A markovian approach for arabic root extraction." *Int. Arab J. Inf. Technol.*, 2011.

[65] Y. Alhanini and M. J. Ab Aziz, "The enhancement of arabic stemming by using light stemming and dictionary-based stemming," *JSEA*, 2011.

[66] K. Elghamry, "A constraint-based algorithm for the identification of arabic roots," in *Proceedings of the Midwest Computational Linguistics Colloquium. Indiana University. Bloomington, IN*, 2005.

[67] P. Rodrigues and D. Cavar, "Learning arabic morphology using statistical constraint-satisfaction models," *Amsterdam studies in the theory and history of linguistic science. Series 4*, 2007.

[68] Y. Choueka, "Mlim-a system for full, exact, on-line grammatical analysis of modern hebrew," in *ICCE*, 1990.

[69] Q. Yaseen and I. Hmeidi, "Extracting the roots of arabic words without removing affixes," *Journal of Information Science*, 2014.

[70] E. Daya, D. Roth, and S. Wintner, "Learning hebrew roots: Machine learning with linguistic constraints," in *EMNLP*, 2004.

[71] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *NIPS*, 2014.

[72] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421, 2015.

[73] D. Gerz, I. Vulić, E. Ponti, J. Naradowsky, R. Reichart, and A. Korhonen, "Language modeling for morphologically rich languages: Character-aware modeling for word-level prediction," *TACL*, 2018.

[74] Y. Zhang, Z. Ye, Y. Feng, D. Zhao, and R. Yan, "A constrained sequence-to-sequence neural model for sentence simplification," *arXiv preprint arXiv:1704.02312*, 2017.

[75] P. Anderson, B. Fernando, M. Johnson, and S. Gould, "Guided open vocabulary image captioning with constrained beam search," in *EMNLP*, 2017.

[76] M. M. al Zabidi and S. M. Murthada, "Taj al-'arus min jawahir al-qamus," *Dar al-Hidayah*, 1886.

[77] M. Elrazzaz, S. Elbassuoni, K. Shaban, and C. Helwe, "Methodical evaluation of arabic word embeddings," in *ACL*, 2017.

[78] A. Freitas, S. Barzegar, J. E. Sales, S. Handschuh, and B. Davis, *Semantic Relatedness for All (Languages): A Comparative Analysis of Multilingual Semantic Relatedness Using Machine Translation*, 2016.

[79] J. Chang, J. Boyd-Graber, C. Wang, S. Gerrish, and D. M. Blei, "Reading tea leaves: How humans interpret topic models," in *NIPS*, 2009, pp. 288–296.

[80] X. Wang, A. McCallum, and X. Wei, "Topical n-grams: Phrase and topic discovery, with an application to information retrieval," in *ICDM*. IEEE, 2007, pp. 697–702.

[81] R. V. Lindsey, W. P. Headden III, and M. J. Stipicevic, "A phrase-discovering topic model using hierarchical pitman-yor processes," in *Proc. EMNLP-CoNLL*, 2012, pp. 214–222.

[82] D. M. Blei and J. D. Lafferty, "Visualizing topics with multi-word expressions," *arXiv preprint arXiv:0907.1013*, 2009.

[83] M. Danilevsky, C. Wang, N. Desai, J. Guo, and J. Han, "Automatic construction and ranking of topical keyphrases on collections of short documents," in *SDM*, 2014.

[84] H. M. Wallach, "Topic modeling: beyond bag-of-words," in *Proc. ICML*. ACM, 2006, pp. 977–984.

[85] W. X. Zhao, J. Jiang, J. He, Y. Song, P. Achananuparp, E.-P. Lim, and X. Li, "Topical keyphrase extraction from twitter," in *Proc. ACL*, 2011, pp. 379–388.

[86] H. D. Kim, D. H. Park, Y. Lu, and C. Zhai, "Enriching text representation with frequent pattern mining for probabilistic topic modeling," *Proceedings of the American Society for Information Science and Technology*, vol. 49, no. 1, pp. 1–10, 2012.

[87] A. Gruber, Y. Weiss, and M. Rosen-Zvi, "Hidden topic markov models," in *Proc. AISTAT*, 2007, pp. 163–170.

[88] Y. Jo and A. H. Oh, "Aspect and sentiment unification model for online review analysis," in *Proc. WSDM*. ACM, 2011, pp. 815–824.

[89] K. S. Hasan and V. Ng, "Conundrums in unsupervised keyphrase extraction: making sense of the state-of-the-art," in *Proc. ACL*, 2010, pp. 365–373.

[90] M. Halliday, "Lexis as a linguistic level," *In memory of JR Firth*, pp. 148–162, 1966.

[91] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proc. VLDB*, vol. 1215, 1994, pp. 487–499.

[92] J. Han, M. Kamber, and J. Pei, *Data mining: concepts and techniques*. 3rd ed., Morgan Kaufmann, 2011.

[93] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *ACM SIGMOD Record*, vol. 29. ACM, 2000, pp. 1–12.

[94] R. Mihalcea and P. Tarau, "Textrank: Bringing order into texts," in *Proceedings of EMNLP*. Barcelona, Spain, 2004, p. 275.

[95] Z. Liu, P. Li, Y. Zheng, and M. Sun, "Clustering to find exemplar terms for keyphrase extraction," in *Proc. EMNLP*. ACL, 2009, pp. 257–266.

[96] K. Church, W. Gale, P. Hanks, and D. Kindle, "chapter 6. using statistics in lexical analysis," *Using statistics in lexical analysis*, p. 115, 1991.

[97] T. Pedersen, "Fishing for exactness," *arXiv preprint cmp-lg/9608010*, 1996.

[98] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *JRML*, vol. 3, pp. 993–1022, 2003.

[99] Z. Ma, X. Xie, and Z. Geng, "Structural learning of chain graphs via decomposition," *JMLR*, vol. 9, p. 2847, 2008.

[100] T. Griffiths, "Gibbs sampling in the generative model of latent dirichlet allocation," *Standford University*, vol. 518, no. 11, pp. 1–3, 2002.

[101] T. Minka, "Estimating a dirichlet distribution," Technical report, M.I.T., 2000.

[102] Q. Mei, X. Shen, and C. Zhai, "Automatic labeling of multinomial topic models," in *Proc. SIGKDD*. ACM, 2007, pp. 490–499.

[103] J. H. Lau, K. Grieser, D. Newman, and T. Baldwin, "Automatic labelling of topic models," in *Proc. ACL*, 2011, pp. 1536–1545.

[104] M. F. Porter, "An algorithm for suffix stripping," *Program: electronic library and information systems*, vol. 14, no. 3, pp. 130–137, 1980.

[105] A. K. McCallum, "Mallet: A machine learning for language toolkit," 2002.

[106] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "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)*, 2019, pp. 4171–4186.

[107] G. Lample and A. Conneau, "Cross-lingual language model pretraining," *arXiv preprint arXiv:1901.07291*, 2019.

[108] F. Guzmán, P.-J. Chen, M. Ott, J. Pino, G. Lample, P. Koehn, V. Chaudhary, and M. Ranzato, "The FLORES evaluation datasets for low-resource machine translation: Nepali–English and Sinhala–English," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019. [Online]. Available: https://www.aclweb.org/anthology/D19-1632 pp. 6100–6113.

[109] P. Resnik, "Mining the web for bilingual text," in *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*. Association for Computational Linguistics, 1999, pp. 527–534.

[110] A. Rafalovitch and R. Dale, "United nations general assembly resolutions: A six-language parallel corpus," in *Proceedings of Machine Translation Summit XII*, August 2009.

[111] M. Ziemski, M. Junczys-Dowmunt, and B. Pouliquen, "The United Nations parallel corpus v1. 0," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, 2016, pp. 3530–3534.

[112] P. Koehn, "Europarl: A parallel corpus for statistical machine translation," in *MT summit*, vol. 5, 2005, pp. 79–86.

[113] D. S. Munteanu and D. Marcu, "Improving machine translation performance by exploiting non-parallel corpora," *Computational Linguistics*, vol. 31, no. 4, pp. 477–504, 2005.

[114] D. S. Munteanu and D. Marcu, "Extracting parallel sub-sentential fragments from non-parallel corpora," in *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2006, pp. 81–88.

[115] R. Udupa, K. Saravanan, A. Kumaran, and J. Jagarlamudi, "Mint: A method for effective and scalable mining of named entity transliterations from large comparable corpora," in *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2009, pp. 799–807.

[116] T.-N.-D. Do, V.-B. Le, B. Bigi, L. Besacier, and E. Castelli, "Mining a comparable text corpus for a vietnamese-french statistical machine translation system," in *Proceedings of the Fourth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, 2009, pp. 165–172.

[117] S. AbduI-Rauf and H. Schwenk, "On the use of comparable corpora to improve smt performance," in *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2009, pp. 16–23.

[118] P. Resnik and N. A. Smith, "The web as a parallel corpus," *Computational Linguistics*, vol. 29, no. 3, pp. 349–380, 2003.

[119] J. Chen and J.-Y. Nie, "Parallel web text mining for cross-language ir," in *Content-Based Multimedia Information Access-Volume 1*. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE, 2000, pp. 62–77.

[120] C. Buck and P. Koehn, "Findings of the wmt 2016 bilingual document alignment shared task," in *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, 2016, pp. 554–563.

[121] A. A. Dara and Y.-C. Lin, "Yoda system for wmt16 shared task: Bilingual document alignment," in *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, 2016, pp. 679–684.

[122] L. Gomes and G. P. Lopes, "First steps towards coverage-based document alignment," in *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, 2016, pp. 697–702.

[123] M. Guo, Y. Yang, K. Stevens, D. Cer, H. Ge, Y.-h. Sung, B. Strope, and R. Kurzweil, "Hierarchical document encoder for parallel corpus mining," in *Proceedings of the Fourth Conference on Machine Translation*. Florence, Italy: Association for Computational Linguistics, August 2019. [Online]. Available: http://www.aclweb.org/anthology/W19-5207 pp. 64–72.

[124] C. Buck, K. Heafield, and B. van Ooyen, "N-gram counts and language models from the common crawl," in *Proceedings of the Language Resources and Evaluation Conference*, Reykjavík, Iceland, May 2014. [Online]. Available: https://kheafield.com/papers/stanford/crawl_paper.pdf

[125] J. R. Smith, H. Saint-Amand, M. Plamada, P. Koehn, C. Callison-Burch, and A. Lopez, "Dirt cheap web-scale parallel text from the common crawl," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2013, pp. 1374–1383.

[126] K. Krippendorff, "Computing Krippendorff's alpha-reliability," University of Pennsylvania, Annenberg School for Communication, Tech. Rep., 2011.

[127] P. Koehn, H. Khayrallah, K. Heafield, and M. L. Forcada, "Findings of the wmt 2018 shared task on parallel corpus filtering," in *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*. Belgium, Brussels: Association for Computational Linguistics, 2018, pp. 726–739.

[128] P. Koehn, F. Guzmán, V. Chaudhary, and J. Pino, "Findings of the WMT 2019 shared task on parallel corpus filtering for low-resource conditions," in *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*. Florence, Italy: Association for Computational Linguistics, Aug. 2019. [Online]. Available: https://www.aclweb.org/anthology/W19-5404 pp. 54–72.

[129] H. Schwenk, V. Chaudhary, S. Sun, H. Gong, and F. Guzmán, "Wikimatrix: Mining 135m parallel sentences in 1620 language pairs from wikipedia," *arXiv preprint arXiv:1907.05791*, 2019.

[130] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst, "Moses: Open source toolkit for statistical machine translation," in *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*. Prague, Czech Republic: Association for Computational Linguistics, June 2007. [Online]. Available: https://www.aclweb.org/anthology/P07-2045 pp. 177–180.

[131] H. Schwenk, "Filtering and mining parallel data in a joint multilingual space," *arXiv preprint arXiv:1805.09822*, 2018.

[132] V. Chaudhary, Y. Tang, F. Guzmán, H. Schwenk, and P. Koehn, "Low-resource corpus filtering using multilingual sentence embeddings," in *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*. Florence, Italy: Association for Computational Linguistics, August 2019. [Online]. Available: http://www.aclweb.org/anthology/W19-5435 pp. 263–268.

[133] M. Artetxe and H. Schwenk, "Margin-based parallel corpus mining with multilingual sentence embeddings," *arXiv preprint arXiv:1811.01136*, 2018.

[134] T. Kudo and J. Richardson, "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*, 2018, pp. 66–71.

[135] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A fast, extensible toolkit for sequence modeling," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, 2019, pp. 48–53.

[136] Y. Qi, D. Sachan, M. Felix, S. Padmanabhan, and G. Neubig, "When and why are pretrained word embeddings useful for neural machine translation?" in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 2018, pp. 529–535.

[137] M. Artetxe and H. Schwenk, "Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 597–610, 2019.

[138] C. Buck and P. Koehn, "Quick and reliable document alignment via tf/idf-weighted cosine distance," in *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, 2016, pp. 672–678.

[139] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the society for industrial and applied mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[140] P. Fung and L. Y. Yee, "An ir approach for translating new words from nonparallel, comparable texts," in *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*, 1998.

[141] R. Rapp, "Automatic identification of word translations from unrelated english and german corpora," in *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*. Association for Computational Linguistics, 1999, pp. 519–526.

[142] P. Koehn, "Europarl: A multilingual corpus for evaluation of machine translation," 2002.

[143] V. Shchukin, D. Khristich, and I. Galinskaya, "Word clustering approach to bilingual document alignment (wmt 2016 shared task)," in *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, 2016, pp. 740–744.

[144] M. Medveď, M. Jakubícek, and V. Kovár, "English-french document alignment based on keywords and statistical translation," in *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, 2016, pp. 728–732.

[145] L. Jakubina and P. Langlais, "Bad luc@ wmt 2016: a bilingual document alignment platform based on lucene," in *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, 2016, pp. 703–709.

[146] M. Esplà-Gomis, M. Forcada, S. O. Rojas, and J. Ferrández-Tordera, "Bitextor's participation in wmt'16: shared task on document alignment," in *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, 2016, pp. 685–691.

[147] V. Papavassiliou, P. Prokopidis, and S. Piperidis, "The ilsp/arc submission to the wmt 2016 bilingual document alignment shared task," in *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, 2016, pp. 733–739.

[148] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, "From word embeddings to document distances," in *International conference on machine learning*, 2015, pp. 957–966.

[149] G. Huang, C. Guo, M. J. Kusner, Y. Sun, F. Sha, and K. Q. Weinberger, "Supervised word mover's distance," in *Advances in Neural Information Processing Systems*, 2016, pp. 4862–4870.

[150] K. Atasu, T. Parnell, C. Dünner, M. Sifalakis, H. Pozidis, V. Vasileiadis, M. Vlachos, C. Berrospi, and A. Labbi, "Linear-complexity relaxed word mover's distance with gpu acceleration," in *Proceedings of the 2017 IEEE International Conference on Big Data (Big Data'17)*. IEEE, 2017, pp. 889–896.

[151] G. Balikas, C. Laclau, I. Redko, and M.-R. Amini, "Cross-lingual document retrieval using regularized wasserstein distance," in *European Conference on Information Retrieval*. Springer, 2018, pp. 398–410.

[152] E. Clark, A. Celikyilmaz, and N. A. Smith, "Sentence mover's similarity: Automatic evaluation for multi-sentence texts," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 2748–2760.

[153] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[154] C. Espana-Bonet, A. C. Varga, A. Barrón-Cedeno, and J. van Genabith, "An empirical analysis of nmt-derived interlingual embeddings and their use in parallel sentence identification," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1340–1350, 2017.

[155] M. Guo, Q. Shen, Y. Yang, H. Ge, D. Cer, G. H. Abrego, K. Stevens, N. Constant, Y.-H. Sung, and B. Strope, "Effective parallel corpus mining using bilingual sentence embeddings," *arXiv preprint arXiv:1807.11906*, 2018.

[156] Y. Yang, G. H. Abrego, S. Yuan, M. Guo, Q. Shen, D. Cer, Y.-h. Sung, B. Strope, and R. Kurzweil, "Improving multilingual sentence embedding using bi-directional dual encoder with additive margin softmax," *arXiv preprint arXiv:1902.08564*, 2019.

[157] Y. Rubner, C. Tomasi, and L. J. Guibas, "A metric for distributions with applications to image databases," in *Proceedings of the Sixth International Conference on Computer Vision (ICCV'98)*, 1998, pp. 59–66.

[158] S. Robertson, "Understanding inverse document frequency: on theoretical arguments for idf," *Journal of documentation*, vol. 60, no. 5, pp. 503–520, 2004.

[159] J. Ramos, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, vol. 242. Piscataway, NJ, 2003, pp. 133–142.

[160] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938.

[161] X. Ren, A. El-Kishky, C. Wang, F. Tao, C. R. Voss, and J. Han, "Clustype: Effective entity recognition and typing by relation phrase-based clustering," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 995–1004.

[162] H. Kim, A. El-Kishky, X. Ren, and J. Han, "Mining news events from comparable news corpora: A multi-attribute proximity network modeling approach," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 105–114.

[163] J. Han, C. Wang, and A. El-Kishky, "Bringing structure to text: mining phrases, entities, topics, and hierarchies," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 1968–1968.