Language-Independent Tokenisation Rivals Language-Specific Tokenisation for Word Similarity Prediction

Danushka Bollegala[*,†], Ryuichi Kiryo[†], Kosuke Tsujino[†], Haruki Yukawa[†]
University of Liverpool[*], Amazon[†]
{danubol,kiryor,kosutu,hyukawa}@amazon.com

Abstract

Language-independent tokenisation (LIT) methods that do not require labelled language resources or lexicons have recently gained popularity because of their applicability in resource-poor languages. Moreover, they compactly represent a language using a fixed size vocabulary and can efficiently handle unseen or rare words. On the other hand, language-specific tokenisation (LST) methods have a long and established history, and are developed using carefully created lexicons and training resources. Unlike subtokens produced by LIT methods, LST methods produce valid morphological subwords. Despite the contrasting trade-offs between LIT vs. LST methods, their performance on downstream NLP tasks remain unclear. In this paper, we empirically compare the two approaches using semantic similarity measurement as an evaluation task across a diverse set of languages. Our experimental results covering eight languages show that LST consistently outperforms LIT when the vocabulary size is large, but LIT can produce comparable or better results than LST in many languages with comparatively smaller (i.e. less than 100K words) vocabulary sizes, encouraging the use of LIT when language-specific resources are unavailable, incomplete or a smaller model is required. Moreover, we find that smoothed inverse frequency (SIF) to be an accurate method to create word embeddings from subword embeddings for multilingual semantic similarity prediction tasks. Further analysis of the nearest neighbours of tokens show that semantically and syntactically related tokens are closely embedded in subword embedding spaces.

Keywords: Subtokenisation, Byte Pair Encoding, Language Independent Tokenisation

1. Introduction

One of the first steps in many NLP pipelines is tokenisation – the process of splitting a given text into a sequence of continuous lexical units for the purpose of representing the given text. Tokenisation can be performed at various granularities such as at phrase-level, word-level, sub-word level or character-level, considering the level of textual representation required for a particular task (Riedl and Biemann, 2018). For example, for information retrieval, we must ensure that both documents and user queries are tokenised in a consistent manner considering the relevance of the search results. In specialised domains such as biomedical, proper tokenisation can significantly improve the retrieval accuracy by up to 80% (Jiang and Zhai, 2007). Exceedingly finer tokenisation is likely to return many irrelevant results with incorrect or partial matches, whereas not tokenising larger phrases will return zero results. In this paper, we use the term token to refer to both words as well as subwords, which might not necessarily be morphological units but character n-grams. For example, given the string "Hello_world", where "_" denotes the space character, a possible sequence of subtokens could be H/el/l/o/_/world. As can be seen from this example, some of the subtokens such as H, el,, l are not valid English words, whereas some such as world are. Therefore, the effect of subtokenisation on downstream NLP tasks that require the semantics of the original input string to be retained remains unclear.

The complexity of the tokenisation problem is language dependent. For example, punctuation rules, delimiter characters etc. have found to be adequate to tokenise non-agglutinative languages such as English or Italian (Moreau and Vogel, 2018), whereas non-white space delimited languages such as Japanese or Chinese require more sophisticated methods that jointly perform Part of Speech (PoS) tagging with tokenisation (Kudo et al., 2004). Moreover, hyphenated words, acronyms that use punctuations must be treated as single tokens in most NLP applications, which makes tokenisation a complex problem.

Tokenisation methods can be classified into language-specific tokenisation (LST) and language-independent tokenisation (LIT). LST methods require lexicons for the language under consideration and are often trained on manually tokenised corpora. The accuracy of LST depends on the coverage and quality of the linguistic resources used to train them. In particular, when the coverage of the training resources are poor such as for rare words, named entities or neologisms, the accuracy of tokenisation of out of vocabulary (OOV) words can be low. LST methods have been trained using different sequence labelling methods such as hidden Markov models (HMMs) (Jurish and Würzner, 2013), conditional random fields (CRFs) (Kudo et al., 2004) and recurrent neural networks (RNNs) (Morita et al., 2015).

LIT has gained popularity as an alternative to LST (Sennrich et al., 2016; Zhu et al., 2019; Kudo and Richardson, 2018; Kudo, 2018; Schuster and Nakajima, 2012) because, unlike LST, LIT methods do not require predefined vocabularies nor manually tokenised texts, and operate on statistical information obtained from a large text corpora. For example, text compression methods such as byte pair encoding (BPE) (Gage,

1994; Sennrich et al., 2016) and language modelling (LM) methods (Kudo, 2018) automatically select frequent subwords as tokens, and segment a given text such that some loss function (e.g. negative likelihood or code length) is minimised. LIT has become the de-facto standard in text generation applications such as Neural Machine Translation (NMT) (Ataman and Federico, 2018), where a small vocabulary size is preferred for speeding up the decoding process (Bahdanau et al., 2015). Moreover, subword regularisation using probabilities produced by the LM approach has shown to improve the accuracy of NMT (Kudo, 2018). However, as seen from our previous example, unlike LST, LIT often produces nonsensical subwords, which are not valid morphological units (Zhu et al., 2019).

As discussed above, LST and LIT have complementary trade-offs. It remains unclear whether the loss in morphological information and the noise introduced by LIT out weights the benefits of using a small and fixed vocabulary, enabling us to overcome OOV issues across typologically diverse languages. To empirically answer this question, we compare LST vs. LIT for multilingual lexical semantic similarity prediction for the eight languages: English (en), German (de), Spanish (es), Farsi (fa), Italian (it), Japanese (ja), Turkish (tr) and Thai (th). Our contributions and findings in this paper can be summarised as follows:

- We independently conduct LST and LIT on eight languages and use Global Vectors (GloVe) (Pennington et al., 2014) to learn word embeddings. We then predict the semantic similarity between two words using the learnt word embeddings, and measure the correlation against human similarity ratings across a suite of benchmark datasets.
- We evaluate different methods to compose word embeddings from subword embeddings and find that Smoothed Inverse Frequency (SIF) (Arora et al., 2017) to outperform simple averaging, which has shown to be a strong baseline in prior work.
- For LIT methods, for the first time, we compare BPE and LM in terms of both tokenisation speed and their accuracies for predicting semantic similarity between words.
- Our experimental results show that for smaller (less than 100K tokens) vocabularies, LIT consistently outperforms LST. Moreover, between LIT methods, LM outperforms BPE.

Our goal in this paper is not to propose novel methods for LST or LIT. Instead, our objective is to compare LST and LIT for word embedding learning, and empirically evaluate the differences across a diverse set of languages using semantic similarity prediction as an evaluation task. Tokenisation is one of the fundamental pre-processing steps in any NLP pipeline and has a long and established history of numerous approaches. Although we cannot hope to conduct an extensive survey of all prior tokenisation methods due to space limitations, we briefly summarise the background details of LIT and LST methods in Section 2. to support the readers to understand the experimental results described in the paper. Several prior work have already investigated the effect of subtokenisation for different NLP tasks. We describe these related prior work in Section 3. and highlight the important differences between the findings reported in this paper. Evaluation protocol and experimental results comparing LST vs. LIT methods are described in Section 4..

2. Background

2.1. Language Specific Tokenisation

LST methods use language-specific resources such as lexicons, manually tokenised corpora and/or language-specific rules. Earlier versions of the Stanford Core NLP toolkit (Manning et al., 2014) internally used JFLex[1], a meta language for specifying tokenisation rules based on regular expressions and procedures, to execute when a rule matches. Unlike the statistical tokenisers, rule-based tokenisers are easier to debug and their behaviour is deterministic. For example, a product name might be required to tokenise in a specific manner, which is easier to specify as a rule rather than having to prepare numerous manually tokenised examples of contexts to train a model. For those reasons, rule-based tokenisers have been used extensively in industrial NLP applications either as a standalone module or in conjunction with statistical tokenisers (Remus et al., 2016).

Statistical or machine learning-based tokenisation methods model tokenisation as a sequence labelling problem where we must predict whether a token boundary must be placed at a given position in an input text string. For example, information about the current token and its context such as previous or following tokens can be used as features for training a sequence labeller such as a hidden Markov model (Papageorgiou, 1994), conditional random field (Kudo et al., 2004) or a recurrent neural network (Chen et al., 2015). In languages such as Japanese or Chinese where multiple possible tokenisations of the input string exist, one must find the most likely sequence of tokens (Kudo et al., 2004). This can be modelled as a dynamic programming problem and solved efficiently via forward-backward inference methods. Moreover, token boundaries as well as morphological properties of the tokens such as their part-of-speech (POS) tags can be simultaneously determined, which is known as morphological analysis.

To train statistical tokenisers we need lexicons, which lists all the words in a language, and manually tokenised texts as the training data. Words that do not occur in the lexicon (i.e. out of vocabulary words) can get incorrectly tokenised and is a major cause of errors in statistical tokenisers. Moreover, manually tokenised texts might not be available for the domain in which we might want to use the tokeniser after training, and

[1] https://jflex.de/

manually creating such labelled training data can be both costly as well as time consuming.

2.2. Language Independent Tokenisation

Tokenising texts into subwords/subtokens, lexical units smaller than words/tokens, has received much attention lately with their effectiveness in deep learning-based NLP models. For example, named entities, cognates/loanwords, and morphologically complex words that contain multiple morphemes are extremely challenging to properly tokenise because the occurrences of such terms are rare even in large training datasets. On the other hand, substrings of such terms are likely to be more frequent. Tokenising texts into subtokens has been sufficient for a broad range of NLP tasks such as machine translation (Sennrich et al., 2016) and language modelling (Pires et al., 2019), where tokens are represented using lower-dimensional embedding vectors and fed into deep learning architectures.

Sennrich et al. (2016) proposed a subtokenisation method inspired by BPE, which is a data compression technique that iteratively replaces the most frequent pair of bytes in a sequence with a single unused byte. Specifically, the set of symbols (i.e. symbol vocabulary) is initialised with the set of characters and each word is represented as a sequence of characters, plus a special end-of-word symbol. This is useful if we want to restore the original tokenisation after subtokenising. Next, BPE iteratively counts all symbol pairs and replaces each occurrence of the most frequent pair ('A', 'B') with a new symbol 'AB'. Each merge operation produces a new symbol, which represents a character n-gram. Frequent character n-grams (or whole words) are eventually merged into a single symbol. Because of this bottom-up nature of BPE, it does not require a shortlist and the final symbol vocabulary size is equal to the size of the initial vocabulary, plus the number of merge operations. This is ideal for producing smaller vocabularies in natural language generation tasks such as machine translation to reduce the GPU memory footprints and the training time because every element in the output vocabulary is a potential candidate for generation. The number of merge operation is a hyperparameter in BPE that can be tuned to generate arbitrarily smaller vocabulary sizes.

An alternative subtokenisation method was proposed by Kudo (2018) based on the unigram language model under the assumption that each subword occurs independently, and consequently, the probability of a subword sequence can be computed as the product of the individual subword occurrence probabilities. This method iteratively increases the size of the vocabulary (set of subtokens) such that a user-defined limit is reached. It computes the optimal set of subtokens based on their occurrence probabilities, estimated using the expectation maximisation (EM) algorithm. The initial seed vocabulary can be set to the union of all characters and the most frequent substrings in the corpus. Because the vocabulary contains all individual characters in the corpus, subtokenisation using the unigram language model produces a probabilistic mixture of characters, subtokens and word segmentations.

Both BPE and unigram language model can be trained using untokenised text corpora. Moreover, both methods can be used independently of the language, which make them ideal candidates for tokenising resource poor languages. Because of those reasons BPE and unigram language model are considered as LIT methods to compare in this paper.

3. Related Work

Learning embeddings for the subtokens produced by LIT methods has shown to be an effective method to overcome data sparseness issues encountered when training named entity recognisers for low-resource languages such as Uyghur and Bengali (Chaudhary et al., 2018). By modelling a word as a bag of subtokens and combining pre-trained subtoken embeddings to represent rare out-of-vocabulary (OOV) words, Zhao et al. (2018) obtained SoTA results for joint prediction of POS tagging and morphosyntactic attributes in 23 languages. These prior work show that LIT can be used to overcome OOV and rare word related issues and is especially effective for resource poor languages, but did not perform a systematic comparison between LIT vs LST methods for those tasks.

Zhu et al. (2019) compared supervised morphological segmentation (SMS) by CHIPMUNK, Morfessor (a family of generative probabilistic models for unsupervised morphological segmentation) and BPE. They train word and subword embeddings using skip-gram with negative sampling (SGNS) (Mikolov et al., 2013a). They used multilingual word similarity, universal dependency parsing and fine-grained entity typing as the evaluation tasks. They found that subword SGNS embeddings outperform subword-agnostic SGNS embeddings for morphologically richer languages such as Finnish and Turkish. SMS, which is trained according to the readily available gold standard morphological segmentations, performs best for word similarity but worst for entity typing. Compared to BPE, which produces short and nonsensical subwords, Morfessor is a conservative segmenter that captures longer subwords. Consequently, Morfessor reports the best performance on entity typing. More importantly they emphasise that there is no single configuration that outperforms the others in all three tasks, which demonstrates the challenges involved in using subword information in a consistent manner across languages and tasks. Moreover, addition, elementwise multiplication of subword embeddings, and self-attention are used as the composition functions for creating word embeddings from subword embeddings. They found that addition to be an extremely robust composition function across languages and tasks. Surprisingly, the more sophisticated self-attention reports poor performance in many tasks. In this paper, we propose the use of smoothed inverse frequency (SIF), which was originally proposed by Arora et al. (2017) for creating sentence embeddings from word embeddings, for

the purpose creating word embeddings from subword embeddings.

4. Evaluation Protocol

Evaluating tokenisation methods is a challenging task because there is no universally agreed gold standard for tokenisation (Habert et al., 1998; Webster and Kit, 1992). Tokenisation depends both on the language as well as the task for which it is used. Although there are some manually tokenised texts such as the Penn Treebank dataset (Marcus et al., 1994) for English and Kyoto University corpus (Kawahara et al., 2002) for Japanese that can be used to train and evaluate LST methods, no such resources are available for LIT evaluation. Indeed, given that the subtokens produced by LIT methods are arbitrary and depends on the size of the vocabulary specified by the user and the statistics in the corpus used to train the LIT method, what is a valid LIT of a given text remains undefined in the first place. Therefore, following prior work comparing LST and LIT methods, we resort to an extrinsic evaluation approach where we use the tokenised output produced by a particular tokenisation method to solve an NLP task and measure its performance.

To evaluate the ability of LST and LIT methods for producing semantically meaningful tokens, we first tokenise a given text corpus using a particular tokenisation method and then use a word embedding learning method to learn embeddings for the generated tokens. Next, we use the learnt embeddings to compute the similarity between two words and compare that with the similarity ratings assigned by human annotators for those two words. If there exists a high degree of correlation between the predicted similarity scores and the human ratings, then it follows that the tokens produced by the employed tokenisation method correctly preserves the semantic information about words. Semantic similarity prediction has been used as an evaluation task in prior work comparing tokenisation methods (Zhu et al., 2019).

To cover a diverse set of languages with different tokenisation complexities, we select English (en), German (de), Spanish (es), Farsi (fa), Italian (it), Japanese (ja), Turkish (tr) and Thai (th). For each of those languages we downloaded the March 2019 Wikipedia dump[2] and used Wikiextractor[3] to extract texts. We then used the Pragmatic Segmenter[4] to split each Wikipedia article into a set of sentences.

For LST, we used spaCy[5] with its corresponding pretrained LST models[6] for en, de, fr, es, it, fa, th and tr. For ja, we used the CRF-based Japanese tokeniser MeCab[7] with the Japanese IPA dictionary (IPAdic) as the backend of spaCy. Table 1 shows the numbers of sentences extracted and the unique tokens for each language.

Language	#sentences	#LST tokens
en	98,382,467	2,441,459,380
de	43,733,620	860,259,675
es	21,824,361	616,392,562
fa	4,334,205	82,277,928
it	16,888,201	489,437,122
ja	19,258,206	547,956,927
tr	4,006,783	62,444,210
th	777,397	40,105,530

Table 1: Sizes of corpora used in the experiments

For LIT, we consider BPE and unigram language modelling (LM) both implemented in sentencepiece[8]. Specifically, we randomly shuffle sentences in each corpus and train LM models with vocabulary sizes of 20K, 50K, 100K and 1M tokens. For BPE, we train models with vocabulary sizes of 20K, 50K and 100K tokens. As discussed later in section 5., training time of BPE is significantly longer compared to that of LM, which prevented us from creating 1M model for BPE. The character coverage rate and maximum sentence length in sentencepiece are set respectively to 1.0 and 16384 to cover 99% of sentences in the corpora.

4.1. Token Embedding

For corpora tokenised by LST and LIT methods, we use GloVe to learn separate token embedding sets for each language. We set the co-occurrence window size to 15 tokens and the frequency threshold ($x_{\max}$) to 100 in our experiments. We trained 100 and 300 dimensional token embeddings and found the latter to perform better in our experiments across languages. Due to the space limitations, we show experimental results only for 300 dimensional embeddings.

A word can be tokenised into multiple subwords by both LST and LIT methods. For the purpose of composing the embedding of a word from the embeddings of its subwords, Zhu et al. (2019) compared vector addition, elementiwse multiplication and self-attention-based composition (Lin et al., 2017). They found vector addition to outperform other composition methods across languages and tasks. On the other hand, prior work on sentence embedding have shown that a weighted-average of word embeddings to produce simple yet surprisingly accurate sentence embeddings (Arora et al., 2017; Ethayarajh, 2018). Inspired by these prior findings, we propose and compare three methods for composing a word embedding from its subword embeddings as follows:

unweighted: This is the simple unweighted vector addition that reported the best performance in Zhu et al. (2019).

weighted: We use the Smoothed Inverse Frequency

[2]https://dumps.wikimedia.org

[3]https://github.com/attardi/wikiextractor

[4]https://github.com/diasks2/pragmatic_segmenter

[5]https://spacy.io/

[6]https://spacy.io/models

[7]https://github.com/taku910/mecab

[8]https://github.com/google/sentencepiece

(SIF) (Arora et al., 2017), where a word embedding $\boldsymbol{w}$ is computed as the sum of its constituent set of subwords, $\mathcal{S}(w)$, weighted by their inverse unigram probabilities, $p(x)$, for subwords $x \in \mathcal{S}(w)$ as given by (1).

$$\boldsymbol{w} = \sum_{x \in \mathcal{S}(w)} \frac{a}{a + p(w)} \boldsymbol{x} \quad (1)$$

Here, the smoothing parameter a is set to 0.001 following Arora et al. (2017).

weighted + PC removal: After creating word embeddings using (1), we substract the first Principal Component (PC) as suggested by Arora et al. (2017) to remove information that is common to all words, thereby emphasising the relative semantic differences among words.

4.2. Datasets and Evaluation Measures

To evaluate the word embeddings created using different tokenisation and composition methods described above, we use the datasets created for en, de, fr, es, it and fa in SemEval 2017 Task 2 (Camacho-Collados et al., 2017) monolingual word similarity evaluation task. For ja we used the dataset created by Kodaira et al. (2016) via crowd sourcing for evaluating lexical simplification rules, which covers word-pairs categorised into different PoS categories. For th, we used Thai SimLex-999 dataset created by Netisopakul et al. (2019). They first translated the word-pairs in the English SimLex-999 (Hill et al., 2015) and then asked 16 annotators, who are native Thai speakers, score the word-pairs for similarity, following the guidelines of SimLex-999 (Hill et al., 2015). For tr, we used the AnlamVer dataset (Ercan and Yıldız, 2018) contains relatedness and similarity ratings for 500 Turkish word-pairs, annotated by 12 human annotators. Following the official evaluation measure used in SemEval 2017 Task 2, on all datasets we report the harmonic mean of the Spearman and the Pearson correlation coefficients, computed between human similarity ratings and cosine similarities between the words computed using their subword-composed embeddings.

5. Results

Performances of different tokenisation methods and composition methods across languages are summarised in Table 2. Among the composition methods, we see that weighted+PC removal (SIF) consistently outperforms both unweighted and weighted for all languages. To the best of our knowledge, SIF has not been used before for creating word embeddings from subword embeddings. Ethayarajh (2018) showed that by modifying the random walk model proposed by Arora et al. (2016) such that the probability of generating a word given its discourse is proportional not with the inner-product between embeddings, but with their angular distance, vector length confounding effects in SIF can be rectified to create more accurate sentence embeddings. Given such developments, an interesting future research direction would be to apply sentence embedding methods to learn better word embeddings given a subtokenisation.

From Table 2, we see that the best performances are reported by LST for all languages except for de and fa where respectively LM and BPE are the best. Interestingly, for smaller vocabulary sizes (50K, 100K), we see that LM and BPE outperform LST in each language. Given that LIT methods have been popularly used in NMT, where decoder vocabularies are typically less than 100K, it is encouraging to see that this benefit is transferrable to other NLP tasks such as semantic similarity prediction.

In de and fa where morphological agglutination and partial usage of fusional features are common (e.g. in the case system), we see that LIT methods such as BPE and LM outperform LST. For example, spaCy de tokeniser does not split compounds such as selbstfahrendes (selbst = self, fahren = drive, des = a conjugative suffix), while LM with vocabulary size 100K correctly splits it into selbst/fahren/des. For th, we see that LST trained with vocabulary sizes of 50K and 100K perform better than other settings. On the other hand, LM trained with a vocabulary size of 20K performs comparably to the best LST settings for th. Similar to th, for tr we see that the LIT methods trained with vocabularies of sizes 50K and 100K perform better than other settings. In particular, for tr LIT consistently outperforms LST. This can be explained by the fact that tr being a highly inflectional language with a derivational morphology. This result reinforces the observation that subword tokenisation via LIT methods is particularly effective for strong inflective languages such as ja and tr, when creating word embeddings.

Given the language independent nature of LIT methods, an interesting question is whether it would be beneficial to train a single LIT tokeniser for a group of languages. To address this question,iIn a preliminary study, we mixed all corpora in Table 1 to create a single multilingual corpus and trained LM and BPE on it. However, the tokeniser models obtained by this approach were poor, which suggests that LIT methods must be trained on monolingual corpora. This could be due to the disproportions of the sizes of the corpora available for different languages, which bias the subtoken statistics for some languages than the others. Careful data sampling would be needed to create balanced text corpora for learning universal LIT models. Investigating methods for learning universal LIT models is beyond the scope of the current paper and would be an interesting future research direction.

5.1. Effect of Part-of-Speech

To further study the effective of tokenisation for different POS categories, we use the Japanese word similarity dataset created by Kodaira et al. (2016). This dataset classifies word-pairs according to POS category of the two words being compared. In particular, both words in a word-pair belong to the same POS category, which makes it an ideal candidate for study-

Composition	model	N	de	en	es	fa	it	ja	th	tr
unweighted	LST	50K	34.95	52.89	55.15	50.01	49.62	9.66	55.75	26.81
		100K	48.35	58.90	61.41	50.15	61.01	13.04	55.57	27.06
		1M	50.07	63.29	64.78	50.42	62.42	14.37	52.88	20.27
		10M	54.10	63.80	66.20	50.57	64.88	14.85	54.90	20.52
	LM	20K	52.17	55.61	53.78	58.59	52.89	21.11	35.52	35.34
		50K	60.66	65.05	60.72	59.48	60.91	22.77	31.36	32.51
		100K	63.38	66.55	65.46	59.19	62.29	18.00	32.37	36.47
		1M	59.46	63.06	64.85	58.62	62.71	5.33	35.59	33.51
	BPE	20K	49.41	51.47	52.86	55.09	55.73	18.82	49.56	35.34
		50K	58.33	63.63	59.54	59.33	60.56	13.76	53.98	37.00
		100K	61.33	63.98	62.79	58.60	63.26	14.20	52.86	31.96
weighted	LST	50K	34.80	53.34	55.90	50.18	49.70	13.20	57.01	26.81
		100K	48.08	59.17	62.80	50.58	61.74	13.84	56.84	27.06
		1M	50.08	63.04	67.37	50.41	63.55	21.46	54.07	20.27
		10M	54.01	63.40	68.55	50.57	65.74	22.04	56.06	20.52
	LM	20K	51.80	56.99	54.08	58.05	53.35	23.72	61.43	32.73
		50K	60.4	65.34	61.58	58.49	60.95	27.00	61.03	33.19
		100K	63.93	66.48	66.57	58.25	62.75	26.80	58.18	34.27
		1M	59.85	62.62	66.50	57.83	64.23	20.77	36.15	32.28
	BPE	20K	51.84	51.07	53.56	54.52	55.69	22.95	51.49	32.73
		50K	58.8	63.69	60.30	58.65	61.10	22.25	55.78	35.99
		100K	61.67	64.24	63.91	58.34	63.22	21.90	53.80	29.57
weighted + PC removal	LST	50K	38.90	55.00	59.73	54.52	53.52	19.16	63.58	27.09
		100K	51.82	61.84	67.43	59.23	65.34	23.29	64.69	28.73
		1M	63.12	71.39	75.41	60.92	70.53	30.98	63.81	29.31
		10M	65.61	71.49	74.81	60.01	70.85	30.87	64.85	28.95
	LM	20K	53.79	57.29	57.83	59.84	54.30	25.68	62.49	37.18
		50K	62.45	66.69	65.45	62.72	63.05	28.97	61.92	38.74
		100K	64.56	67.58	71.38	64.20	65.63	29.39	59.33	36.68
		1M	68.14	68.23	74.35	64.26	70.16	22.29	38.47	32.68
	BPE	20K	53.03	52.56	56.36	56.86	55.66	23.89	52.04	37.18
		50K	60.17	64.28	64.24	63.19	62.76	21.93	55.92	41.22
		100K	62.60	65.17	68.75	65.40	65.76	21.80	53.66	28.51

Table 2: Harmonic mean of the Spearman and Pearson correlation coefficients computed between the predicted cosine similarity scores using word embeddings and human similarity ratings for different languages. Best result for each language is bolded.

ing the effect of tokenisation on different POS categories. As observed in Table 2, among the different composition methods, SIF method reported the best results across languages. Therefore, we use SIF for creating word embeddings from subword embeddings in this experiment. Specifically, we use the GloVe embeddings for Japanese subtokens/tokens obtained by a particular tokenisation method and use SIF to create the word embeddings for each word in word-pairs in the Japanese semantic similarity dataset. The similarity between two words is computed by the cosine of the angle between the corresponding word embeddings. Next, we measure the Spearman and Pearson correlation between the predicted similarity scores and the human ratings for each POS category and report the harmonic mean between the Spearman and Pearson correlation coefficients as done in the previous experiment. Arithmetic mean (average) over the four POS categories – adjectives, adverbs, nouns and verbs, are reported in Table 3.

From Table 3, we see that the performance of LST with smaller vocabularies such as 50K or 100K tokens for adjectives is poor. Compared to other POS categories, adjectives are highly inflected in Japanese and depend on the tense of the sentence. Therefore, a smaller vocabulary might not be sufficient to cover all the variants of adjectives. On the other hand, LIT methods such as LM significantly outperforms LST even with a smaller vocabulary size of 20K subtokens. This result reinforces the observation we made in Table 2 that LIT methods are attractive for obtaining good performance with smaller vocabulary sizes. Increasing the size of the vocabulary results in a steady improvement in performance for LST. However, the same cannot be said about LIT. For example, the performance of LM increases when the size of the vocabulary is increased from 20K to 50K but drops when it is increased beyond 50K. This issue is particularly sever for nouns.

Method	N	adjective	adverb	noun	verb	average
LST	50K	7.50	22.80	20.04	26.29	19.16
	100K	6.99	28.19	25.35	32.61	23.28
	1M	24.85	35.67	27.06	36.34	30.98
	10M	24.63	35.66	26.70	36.45	30.86
LM	20K	30.66	21.79	19.85	30.42	25.68
	50K	32.32	23.97	24.54	35.05	28.97
	100K	33.33	26.64	23.87	33.70	29.38
	1M	24.20	22.64	12.30	29.99	22.28
BPE	20K	24.13	22.98	18.40	30.01	23.88
	50K	22.87	19.54	16.66	28.64	21.93
	100K	22.95	19.70	16.12	28.42	21.80

Table 3: Harmonic mean of the Spearman and Pearson correlation coefficients computed between the predicted cosine similarity scores using word embeddings computed using the SIF method and human similarity ratings for Japanese word-pairs. Results are shown separately where both words in a word-pair belongs to a particular POS category. The final column shows the arithmetic mean over the four POS categories adjectives, adverbs, nouns and verbs.

ing	ed
ed (0.610188)	ing (0.610188)
_utiliz (0.416099)	_aggravat (0.3683)
_consolidat (0.4143)	_dispos (0.3682)
_thereby (0.4138)	_encas (0.3670)
_manipulat (0.4125)	_accentuat (0.3666)
_incorporat (0.4029)	_clipp (0.3634)
_facilitat (0.3980)	_precipitat (0.3600)
_expell (0.3937)	_produc (0.3580)
_involves (0.3916)	_exacerbat (0.3576)
_dedicat (0.3895)	_rechristen (0.3543)
_without (0.3841)	_supplant (0.3498)

Table 4: Nearest neighbours and their cosine similarity scores (indicated within brackets) for the two English suffixes ing and ed.

Similar trends can be observed with BPE as well.
Larger vocabularies contain many smaller subtokens and the probability of a given text getting overtokenised into many smaller tokens increases with the size of the vocabulary for LIT. Creating the embedding for a word using embeddings for its subtokens becomes difficult when the word is split into many subtokens, some of which might be too small to retain the semantics of the original word. Recall that SIF method creates word embeddings as the weighted-average of the subtoken embeddings, ignoring the position of the subtoken in the word. Incorporating character-level embeddings via LSTMs has shown to improve performance for named entity recognition tasks (Zhai et al., 2018). Therefore, applying more sophisticated supervised composition methods such as a recurrent neural network might help to create word embeddings from subtoken embeddings under such situations. We defer this line of investigation for future work. We conclude here that the size of the vocabulary is a hyperparameter of LIT methods that must be carefully set considering the performance of the target task.

5.2. Nearest Neighbour Analysis

Given that some subtokens correspond to character n-grams representing morphology such as inflections, it remains an interesting qualitative analysis to study whether such information is encoded in the learnt subword embeddings. We select prefixes or suffixes that have known inflectional roles and compute the cosine similarity between each prefix/suffix and all other tokens in the vocabulary using the unweighted embedding method to find the nearest neighbours in the embedding space. Specifically, we conduct this nearest neighbour analysis for the three languages: English, Japanese and Turkish. English is selected as a language that uses the space character to denote word boundaries, Turkish and Japanese are selected as agglutinative languages, whereas word boundaries are not marked by the space character in Japanese. We use the LIT models obtained using LM with vocabulary sizes 100K, 50K and 50K respectively for English, Japanese and Turkish for finding the nearest neighbours using subword embeddings.
Table 4 shows the nearest neighbours for the suffixes ed and ing, which often inflects verb tense in English. We use an underscore to denote a token boundary corresponding to the space character. From Table 4, we see that verbs that are frequently inflected using those suffixes are ranked at the top as the nearest neighbours, indicating that the relationship between inflective suffixes and verbs is preserved during LIT.
Table 6 shows the nearest neighbours for the Japanese verb ending form masu. We see that various inflections of masu are listed as the top nearest neighbours such as its past tense (mashita), negation (masen) and the volitional form (mashou). We also see that other frequent sentence ending forms such as desu and kudasai are also listed as nearest neighbours. Similar trends have been reported with distributional word-level embeddings, where both semantically similar as well as related/associated words are often found as the nearest neighbours for a given word when the cosine similarity between word embeddings is used as the neighbourhood criterion (Hill et al., 2015; Weeds et al., 2014).
Table 7 shows the nearest neighbours for the Turkish suffixes iyor and miyor, which respectively denote the present tense and its negation. Likewise in English and Japanese results, we see related words are listed as the nearest neighbours for those suffixes. However, the nearest neighbours retrieved in the case of Turkish are more noisier compared to that for English and Japanese. We believe this is due to the comparatively smaller corpora used for Turkish.
Word embedding spaces learnt by word2vec and GloVe have shown to demonstrate a surprisingly high degree of relational structure, which can be exploited to solve analogies (Allen and Hospedales, 2019; Mikolov et al.,

Analogy	Top candidates
_improving - ing + ed	_improved (0.5817), _improve (0.5791), _prioritize (0.4582), _improvement (0.4526)
_posterior - _prior + _pre	_anterior (0.6705), _dorsal (0.5405), _medial (0.5341), _ventral (0.5266)
_export - _ex + _im	_exports (0.4191), _markets (0.3998), _exporting (0.3906), _importation (0.3850)
- +	(0.6492), (0.6344), (0.5570), (0.5551)
- +	(0.6054), (0.5686), (0.5634), (0.5627)
- +	(0.5424), (0.5046), (0.4738), (0.4677)
meyecek - ecek + iyor	_miyor (0.5093), _miyordu (0.4835), _iyordu (0.4828), _mekteydi (0.4459)
_bunu - nu + na	_buna (0.5207), _rağmen (0.4771), _fakat (0.4610), _ama (0.4530)
_gitmek - mek + ti	_gönderilmiş (0.4135), _gitti (0.4078), ten (0.3550), _yola (0.3424)

Table 5: Top candidates for the analogies ranked according to their cosine similarity (shown within brackets) with the target vector for English, Japanese and Turkish.

(masu)	similarity	info
(mashita)	0.8273	conjugation of masu
(masen)	0.6522	conjugation of masu
(mashou)	0.6400	conjugation of masu
(kudasai)	0.5846	imperative verb for please
(desu)	0.5819	sentence ending
(nasai)	0.5514	imperative verb for do

Table 6: Nearest neighbours and their cosine similarity scores for the Japanese verb masu.

iyor	miyor
iyordu (0.7491)	miyordu (0.6765)
miyor (0.5655)	iyor (0.5655)
ecektir (0.5349)	miyorsa (0.5370)
eceğini (0.5097)	_destekle (0.5130)
eceği (0.5036)	_gel (0.5033)
_geçir (0.5005)	ebiliyordu (0.5012)
iyorum (0.4971)	ememektedir (0.4971)
mektedir (0.4911)	ebiliyor (0.4961)

Table 7: Nearest neighbours and their cosine similarity scores (indicated within brackets) for, iyor and miyor the Turkish suffixes indicating respectively the present tense and its negation.

2013b). To test whether these relational properties exist in subtoken embedding spaces, we use the unweighted word embeddings and solve exemplar analogies as shown in Table 5. Specifically, for an analogy "a is to b as c is to d", given a, b and c we find candidates d that satisfy the analogy according to the cosine similarity between the vector $\boldsymbol{b}-\boldsymbol{a}+\boldsymbol{c}$ and each of the subtoken embedding $\boldsymbol{d}$ in the vocabulary. We then rank the candidates d in the descending order of the cosine similarity scores. The first set of three rows in Table 5 show analogies for English, whereas the second and third sets of three rows respectively show analogies for Japanese and Turkish.

For English we see that suffixes (as in the case for improve) as well as prefixes (as in the case for export) demonstrate a certain level of relational structure in the embedding space. However, analogies are not always correctly preserved in the subtoken embedding spaces as seen from the example for posterior. Although anterior (front of the body) and dorsal (upper side or back of an animal or plant) are closely related to the semantics implied by the resulting vector, they are not perfect candidates. On the other hand, the Japanese subtoken embeddings show interesting analogical structures. For example, subtracting the embedding for the kanji character (mae, meaning before) from (chokuzen, meaning immediately before) and adding (ato, meaning after), we can discover (chokugo, meaning immediately after). We see that the Turkish suffixes ecek (indicating the future tense) and meyecek (indicating the negated future tense) form the analogy meyecek - ecek + iyor with miyor. Overall, we see that the analogical relationships reported for word embeddings in prior work can also be seen with subword embeddings.

5.3. Training time

LIT methods such as BPE and LM must be first trained on an untokenised corpus to compute the vocabularies and the frequencies of the subtokens. Larger corpora that cover various word forms are desirable for this purpose because it enables us to obtain reliable subtoken frequencies for a larger vocabulary. However, the training time depends on the size of the vocabulary and is an important aspect to consider in practice.

In Figure 1, to study the scalability of LIT methods, we compare BPE and LM in a single threaded setting on the same hardware (m5.24xlarge AWS instances) under different numbers of input sentences. To the best of our knowledge, prior work comparing LIT methods have not studied the effect on training time for LM or BPE. From Figure 1, we see that LM is significantly faster than BPE, and its training time decreases with the vocabulary size, while the opposite is true for BPE. This is because BPE iteratively increases the vocabulary until the desired size is reached, whereas LM iteratively decreases the same. Moreover, further speed ups for LM can be easily obtained via multi-threading because the E-step of the likelihood computation in LM is embarrassingly parallelisable.

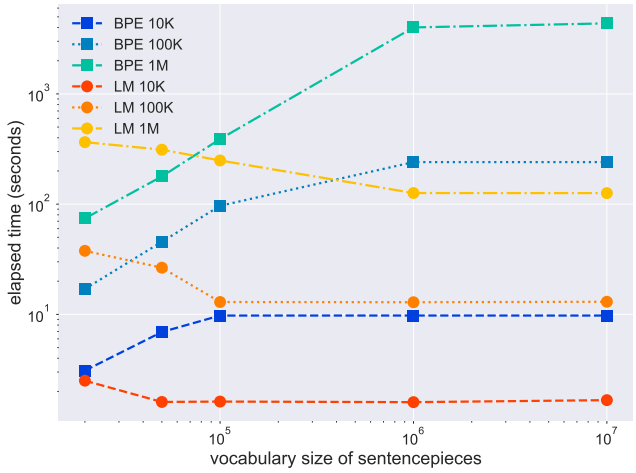

Figure 1: Comparison of the training time for BPE and LM for different numbers of input sentences.

6. Conclusion

We compared LST against two LIT methods (BPE and LM) for multiple languages using similarity prediction as an evaluation task. After tokenising a text corpus, we used GloVe to learn embeddings for the subtokens. Next, we created word embeddings by composing the subtoken embeddings. We used semantic similarity prediction as a evaluation task where we predict the similarity between two words by the cosine of the angle between the corresponding word embeddings. We found that when the vocabulary size is large, LST methods consistently outperform LIT methods. However, for smaller vocabularies (less than 100K), LIT methods outperformed LST methods, suggesting that LIT is suitable for resource poor languages or when smaller models are required. Moreover, SIF method, which weights subword embeddings by unigram probability and subtract the first principal component vector was found to be an effective composition method for creating word embeddings from subword embeddings. We analysed the nearest neighbours for subtokens and found that semantically and syntactically related subtokens are retrieved as the top nearest neighbours using subword embeddings. Moreover, analogical structures, which have been previously reported for word embedding spaces, can also be found even in subword embedding spaces.

7. Bibliographical References

Allen, C. and Hospedales, T. (2019). Analogies explained: Towards understanding word embeddings. In Kamalika Chaudhuri et al., editors, Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pages 223–231, Long Beach, California, USA, 09–15 Jun. PMLR.

Arora, S., Li, Y., Liang, Y., Ma, T., and Risteski, A. (2016). A latent variable model approach to pmi-based word embeddings. Transactions of the Association for Computational Linguistics, 4:385–399.

Arora, S., Liang, Y., and Ma, T. (2017). A simple but tough-to-beat baseline for sentence embeddings. In Proc. of ICLR.

Ataman, D. and Federico, M. (2018). An evaluation of two vocabulary reduction methods for neural machine translation. In Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers), pages 97–110. Association for Machine Translation in the Americas.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. Proc. of ICLR.

Camacho-Collados, J., Pilehvar, M. T., Collier, N., and Navigli, R. (2017). Semeval-2017 task 2: Multilingual and cross-lingual semantic word similarity. In Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), pages 15–26, Vancouver, Canada, August. Association for Computational Linguistics.

Chaudhary, A., Zhou, C., Levin, L., Neubig, G., Mortensen, D. R., and Carbonell, J. (2018). Adapting word embeddings to new languages with morphological and phonological subword representations. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3285–3295. Association for Computational Linguistics.

Chen, X., Qiu, X., Zhu, C., Liu, P., and Huang, X. (2015). Long short-term memory neural networks for Chinese word segmentation. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1197–1206, Lisbon, Portugal, September. Association for Computational Linguistics.

Ercan, G. and Yıldız, O. T. (2018). AnlamVer: Semantic model evaluation dataset for Turkish - word similarity and relatedness. In Proceedings of the 27th International Conference on Computational Linguistics, pages 3819–3836, Santa Fe, New Mexico, USA, August. Association for Computational Linguistics.

Ethayarajh, K. (2018). Unsupervised random walk sentence embeddings: A strong but simple baseline. In Proceedings of The Third Workshop on Representation Learning for NLP, pages 91–100, Melbourne, Australia, July. Association for Computational Linguistics.

Gage, P. (1994). A new algorithm for data compression. C Users J., 12(2):23–38, February.

Habert, B., Adda, G., Adda-Decker, M., de Marëuil, P. B., Ferrari, S., Ferret, O., Illouz, G., and Paroubek, P. (1998). Towards tokenization evaluation. In Proceedings of LREC, volume 98, pages 427–431.

Hill, F., Reichart, R., and Korhonen, A. (2015). Simlex-999: Evaluating semantic models with (gen-

uine) similarity estimation. Computational Linguistics, 41(4):665–695.

Jiang, J. and Zhai, C. (2007). An empirical study of tokenization strategies for biomedical information retrieval. Information Retrieval, 10(4-5):341–363.

Jurish, B. and Würzner, K.-M. (2013). Word and sentence tokenization with hidden markov models. JLCL, 28:61–83, 01.

Kawahara, D., Kurohashi, S., and Hashida, K. (2002). Construction of a japanese relevance-tagged corpus. In Proc. of the 3rd International Conference on Language Resource and Evaluation, pages 2008–2013.

Kodaira, T., Kajiwara, T., and Komachi, M. (2016). Controlled and balanced dataset for Japanese lexical simplification. In Proceedings of the ACL 2016 Student Research Workshop, pages 1–7, Berlin, Germany, August. Association for Computational Linguistics.

Kudo, T. and Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 66–71, Brussels, Belgium, November. Association for Computational Linguistics.

Kudo, T., Yamamoto, K., and Matsumoto, Y. (2004). Applying conditional random fields to Japanese morphological analysis. In EMNLP'04.

Kudo, T. (2018). Subword regularization: Improving neural network translation models with multiple subword candidates. CoRR.

Lin, Z., Feng, M., dos Santos, C. N., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. (2017). A structured self-attentive sentence embedding. In Proc. of ICLR.

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

Marcus, M., Kim, G., Marcinkiewicz, M. A., MacIntyre, R., Bies, A., Ferguson, M., Katz, K., and Schasberger, B. (1994). The penn treebank: Annotating predicate argument structure. In HUMAN LANGUAGE TECHNOLOGY: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994.

Mikolov, T., Chen, K., and Dean, J. (2013a). Efficient estimation of word representation in vector space. In Proc. of International Conference on Learning Representations.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013b). Linguistic regularities in continous space word representations. In Proc. of NAACL-HLT, pages 746 – 751.

Moreau, E. and Vogel, C. (2018). Multilingual word segmentation: Training many language-specific tokenizers smoothly thanks to the universal dependencies corpus. In Proceedings of the 11th Language Resources and Evaluation Conference, Miyazaki, Japan, May. European Language Resource Association.

Morita, H., Kawahara, D., and Kurohashi, S. (2015). Morphological analysis for unsegmented languages using recurrent neural network language model. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2292–2297, Lisbon, Portugal, September. Association for Computational Linguistics.

Netisopakul, P., Wohlgenannt, G., and Pulich, A. (2019). Word Similarity Datasets for Thai: Construction and Evaluation.

Papageorgiou, C. P. (1994). Japanese word segmentation by hidden markov model. In HUMAN LANGUAGE TECHNOLOGY: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: global vectors for word representation. In Proc. of EMNLP, pages 1532–1543.

Pires, T., Schlinger, E., and Garrette, D. (2019). How multilingual is multilingual BERT? In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4996–5001, Florence, Italy, July. Association for Computational Linguistics.

Remus, S., Hintz, G., Biemann, C., Meyer, C. M., Benikova, D., Eckle-Kohler, J., Mieskes, M., and Arnold, T. (2016). EmpiriST: AIPHES - robust tokenization and POS-tagging for different genres. In Proceedings of the 10th Web as Corpus Workshop, pages 106–114, Berlin, August. Association for Computational Linguistics.

Riedl, M. and Biemann, C. (2018). Using semantics for granularities of tokenization. Computational Linguistics, 44(3):483–524, September.

Schuster, M. and Nakajima, K. (2012). Japanese and korean voice search. 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Mar.

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

Webster, J. J. and Kit, C. (1992). Tokenization as the initial phase in nlp. In COLING 1992 Volume 4: The 15th International Conference on Computational Linguistics.

Weeds, J., Clarke, D., Reffin, J., Weir, D., and Keller, B. (2014). Learning to distinguish hypernyms and co-hyponyms. In Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pages 2249–

2259, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Zhai, Z., Nguyen, D. Q., and Verspoor, K. (2018). Comparing CNN and LSTM character-level embeddings in BiLSTM-CRF models for chemical and disease named entity recognition. In Proceedings of the Ninth International Workshop on Health Text Mining and Information Analysis, pages 38–43, Brussels, Belgium, October. Association for Computational Linguistics.

Zhao, J., Mudgal, S., and Liang, Y. (2018). Generalizing word embeddings using bag of subwords. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 601–606. Association for Computational Linguistics.

Zhu, Y., Vulić, I., and Korhonen, A. (2019). A systematic study of leveraging subword information for learning word representations. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 912–932, Minneapolis, Minnesota, June. Association for Computational Linguistics.