# Subword Representations for Modern and Historical Javanese

A thesis presented

by

Nathaniel Clarence Haryanto

Student ID : 1143776

Supervisor : Lea Frermann

to

The Department of Computing and Information Systems

for the subject of

Computer Science Research Project COMP90080 (100 credit points)

for the degree of

Master of Computer Science

The University of Melbourne

Melbourne, Australia

November 2021

# Declaration

I certify that:

- this thesis does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person where due reference is not made in the text.
- where necessary I have received clearance for this research from the University's Ethics Committee and have submitted all required data to the School.
- the thesis is 18,779 words in length (excluding text in images, table, bibliographies and appendices).

Signed: Nathaniel Clarence Haryanto Date: 04/11/2021

# Abstract

Word embeddings is a method to represent words as vectors. This method has been proven to be effective in representing word for various Natural Language Processing (NLP) tasks. However, word embeddings take words for its input which becomes a problem when it comes to *low-resource* language where frequent words might become a rare word due to a lack of large datasets. To substitute word embeddings, subword-based representation usually is preferred for *low-resource* languages. Subword representations are vector representations of subwords used in NLP and more representative for *low-resource* languages which Javanese is a part of compared to word-based representations.

Although there are studies focused on *low-resource* languages, there is no research specifically focusing on the low resource language of historical Javanese. This problem serves as a bottleneck for further development in Javanese, especially NLP and for language preservation. In addition to that, previous research in Javanese use Wikipedia articles as their datasets which contains a lot of loanwords from other languages.

Our dataset consists of documents collected from sastra.org which is a digital library for historical Javanese and Wikipedia documents. Articles from sastra.org are written from 19$^{th}$ and early 20$^{th}$ century and contain little to no loanwords. This means sastra.org articles will be more reliable compared to Wikipedia in terms of language styles and more similar in term of writing style. Documents collected from Wikipedia serve as our modern comparison dataset because they are written after the 2000s. Hence, they contain more subword variations from foreign words and have different writing style.

In this research, we compared different ways to represent words in Javanese by utilizing cross-domain experiments. Based on perplexity scores, subword representations could generalize better compared to its word-based counterparts over different test sets, both in-domain and out-of-domain settings. In addition, we can also conclude that models trained on the Wikipedia dataset could generalize better on both historical and modern test sets than the models trained on Sastra datasets.

In the second experiment, we use several different methods to generalize over noisy inputs, such as Optical Character Recognition (OCR) documents. In this experiment, the subword regularization method, BPE-Dropout, performs best compared to subword-level models and models trained on a synthetically corrupted dataset.

We also analysed word and subword representations qualitatively using nearest neighbours in vector space. Word representations composed from BPE segmentation tend to be morphologically similar to one another because of the agglutinative nature of Javanese. Meanwhile, word embeddings created from baseline and char-3gram tend to be closer semantically.

Ultimately, we are able to deliver our contribution by: (1) developing resources for further research in Javanese, specifically historical Javanese by using subword-based representations; and (2) introducing the Sastra datasets for research in historical Javanese.

# Table of Contents

# List of Figures

# List of Tables

# Acknowledgements

First, I would like to thank my supervisor, Lea Frermann for the guidance and support during the writing of this thesis.

Second, I would like to thank my mentor, Lucia Dwi Krisnawati for the inspiration for this thesis.

Thirdly, I would like to thank my parents for their encouragement past two years.

Lastly, I would like to personally thank Vania & Lopez (2017) for making the source code for their models publicly available and Yayasan Sastra Lestari (2021) for making historical Javanese documents publicly available in their digital library.

# 1. Introduction

Javanese is part of the Austronesian language family closely related to other languages in Indonesia. There are two different types of Javanese: Old Javanese and New Javanese. “Old Javanese” was used between 9th and 15th century which vocabularies are heavily influenced by Sanskrit loanwords. This type of Javanese is no longer used and only preserved in form of poetries and religious texts in Javanese-influenced Bali. Javanese used after 16th century is what historians referred as “New Javanese” whereas “Modern Javanese” refers to Javanese used from 20th century. Aside from Malay and Islamic influence, Modern Javanese also uses a lot of Dutch loanwords in its vocabularies.

Indonesian as a language was officially established in 1928. In recent years following the independence of Indonesia, younger generations tend to prefer Indonesian as the national language for communication. Thus, Javanese linguistic tradition is assumed a linguistic tradition only preserved within ethnic elites that resided in two remaining royal principalities in Central Java before the independence of Indonesia (Errington, 1992).

Despite its wide usage as spoken language, Javanese is included in *critical under-resource* languages (Cieri et al., 2016). Javanese is included in *critical* language because there is an undesirable ratio of supply and demand, typically of teachers and translators (Cieri et al., 2016). Other criterion that Javanese could be included in is *under-resource* language based on digital resource availability. This lack of digital resource also serves as a bottleneck in further research, especially in Javanese and hinders development of Natural Language Processing (NLP) technology, especially for language preservation.

In this research, we use articles written in Modern and New Javanese which are scraped from two different websites: Wikipedia and sastra.org. Wikipedia articles are written recently using Modern Javanese while sastra.org articles are official documents and poetries written in New Javanese from 19th and early 20th century. Since Wikipedia articles are recently written, they tend to have a lot of foreign loanwords including words from English and Indonesian.

Articles from sastra.org were written in Carakan (Javanese script) and transliterated into Latin alphabets by a team of Javanese experts in Indonesia. Sastra.org is managed by Sastra Lestari, a non-profit independent organization in Indonesia aiming to digitalize and transliterate Javanese written scripts (Yayasan Sastra Lestari, 2021). Sastra.org serves as a digital library especially for Javanese literatures. They sourced their transliterated documents from official records saved by government and Dutch colonials. Some of those literatures also ended up in thrift shops, antique shops, or personal collection which are later acquired through legal means and digitalized. The availability of these documents means more opportunity to build a digital resource (e.g., word embeddings) and preserve Javanese as a language.

In Natural Language Processing (NLP), words are often represented using vector representation which is also known as word embeddings. This means words are represented using a numeric value

that captures their meanings in relative to another words. Following its development by Mikolov et al. (2013a), vector representation plays a lot of roles in prediction, translation, and other NLP tasks. It is easy to represent words in *high-resource* languages such as English. However, it becomes a problem when the task is to represent *low-resource* languages like Javanese because some highly used words become rare words in relative to low document resource.

The modern Javanese used in the Wikipedia dataset (Al-Rfou', Perozzi, & Skiena, 2013) is often used to develop new models in recent research. However, the possibility of incorporating historical Javanese into the research has yet to be explored. In this research, we introduce historical Javanese dataset collection (Sastra dataset) as our historical Javanese source. We are interested in understanding how models trained on modern Javanese to generalize over historical Javanese and the other way around. Aside from understanding how different models generalize over different datasets, Javanese subword representations can also be used as initial knowledge for other NLP-related tasks, such as speech recognition or translation.

Recent NLP research (Table 4 Section 2.5) have used subword-based representation for *low-resource* language including Javanese (Gerz, et al., 2018). The subword-based representation now becomes the state-of-art, not just for representing *low-resource* languages, but also *rich-resource* languages. In this research, the subword-based representation is used to tackle the lack of Javanese digital resources.

In addition to *low-resource* languages, subword-based representations could also be used to generalize over noisy/corrupted inputs. Word-based representations assign unknown token representation to noisy inputs because it could not find the representation in the embeddings layer. However, this is not the case for subword-based representation. Because noisy inputs partially match the subwords (Table 3 Section 2.2) in the vocabulary, subword-based representations could create a new representation instead of using unknown tokens representation as an input to language model.

As mentioned previously, the lack of resources in Javanese could be improved is by implementing subword embeddings instead of word embeddings. Instead of taking the whole word as a unit, subword embeddings split the word into smaller parts (morphemes). For example, separate ***ditumbasaken*** ("being bought for") into its root and affixes ***di-, tumbas,*** and ***-aken*** then embed its affixes and root separately. In addition to morphemes, character-based segmentations which will detailed in Section 2.6.4 could also be used as subword units.

The next step is to recombine all parts to form a word embedding by using composition function. This method is useful because (a) each morpheme will be observed more frequently than full words, hence better data for *low-resource* languages and (b) the agglutinative nature of Javanese which means its morphology is regular.

## 1.1. Significance and Implications of the Study

Previous research (Krisnawati & Mahastama, 2018) showed that there is a gap in the research where there were no digital resources for Javanese NLP development. On specific task of Javanese syllabification, Krisnawati & Mahastama (2018) model failed to segment 3-character words and found several inconsistencies in segmenting words, such as *dipungraita* which is segmented into *di.pun.gra.i.ta* instead of *di.pung.ra.i.ta*. Since there is a lack of resources for Javanese, one particular resource which can support syllabification is subword embeddings. We will develop language model (Section 2.8) which will be used to predict most probable word that suits best given the context. Based on previously stated problems, in this research we aim to:

1. Develop resource for further research in Javanese by creating subword-based knowledge using historical and modern data.
2. Introducing Sastra dataset as a resource for reliable historical Javanese dataset.

## 1.2. Research Questions

To achieve our aims, we formulated several questions that we would like to answer in this research:

1. There are several different methods to represent subwords. How do they compare with each other for Javanese?
2. What is the best way to compose subword-based representations?
3. How does each model generalize across modern and historical datasets?
4. How does each model generalize on noisy or corrupted datasets as typically created through Optical Character Recognition (OCR)?

# 2. Literature Review

In this chapter, we will introduce previous studies that are related to this research. First part of this chapter, we will discuss about Javanese as a language, how noisy data is described, and tasks related to domain adaptation in this research (2.1-2.4). Following that, word representations will be discussed. This part will be split into Word Representations, Word Segmentation Methods, and Composition Functions. In those three sections, we will discuss how word representations are formed and how to form a word representation from subwords (Section 2.5-2.7). The final part of this chapter will be discussing about various language models related to this research (Section 2.8).

## 2.1. Javanese as a Language

Javanese is part of the Austronesian language family closely related to languages in Indonesia. Javanese itself is divided into Old and New Javanese; the former now only survives in poetries and religious purposes in Javanese-influenced Bali. New Javanese is the main literary form of Javanese since the 16th century. The written tradition of New Javanese was later preserved by Surakarta and Yogyakarta Sultanate, and later became the basis of Javanese written standards (Ogloblin, 2006). Javanese books were printed in the early 19$^{th}$ century using Javanese script, also known as Carakan. Latin alphabet is later used to replace Javanese script in documents. The later version of New Javanese, also known as Modern Javanese, were used since the 20$^{th}$ century (Ogloblin, 2006) and contains Indonesian, Dutch, and English loanwords.

According to Wedhawati & Arifin (2006), New Javanese contains stratification of Javanese that were not observed in Old Javanese. New Javanese is divided into three styles depending on social context: *Ngoko*, *Madya*, and *Krama*. *Ngoko* is also known as informal speech, usually used to address people with lower status or siblings; *Madya* is used to address people with unknown status; and *Krama* which is usually used to address people with higher status and official style for public announcements, speeches, and documents. Wikipedia articles tend to use informal speech (*Ngoko*) in addition to foreign loanwords, while documents from sastra.org use formal speech (*Krama*). Table 1 shows different styles of Javanese used in Wikipedia and sastra.org.

*Table 1 Differences between Javanese language styles used in Wikipedia and sastra.org*

| **Language** | | |
|---|---|---|
| **Javanese** | Wikipedia (*Ngoko*) | *Wingi* (time expression) *Bapak* (Subject) *lungo* (Verb) *menyang* (preposition) Jakarta (Object). |
| | Sastra.org (*Krama*) | *Kolowingi* (time expression) *Rama* (Subject) *tindak* (Verb) *dhateng* (preposition) Jakarta (Object). |
| **English translation** | | Yesterday (time expression), father (Subject) went (Verb) to (preposition) Jakarta (Object). |

Just like other Austronesian languages, Javanese is also a part of agglutinative language morphologically. Agglutinative language is a language morphology where base words are modified

with one or several affixes. For example, in Javanese *waca* (to read; active) combined with prefix *di-* (passive identifier) becomes *diwaca* (object "is eaten by" subject). Suharno (1982) further describes the usage of affixes in Javanese which are split between verbs and nouns. As observed in Table 2, prefix *m-* changes root word into its active voice, *di-* into its passive voice, while the suffix *-an* changes the root word into its noun form. Based on that, Javanese as a part of agglutinative languages generally are very regular and rarely have irregular verbs. This property makes them particularly amenable to subword modelling.

*Table 2 Javanese root and affixes example.*

| **Language** | | | | | |
|---|---|---|---|---|---|
| **Javanese** | **Wikipedia (*Ngoko*)** | waca | maca | diwaca | wacan |
| | **Subword** | waca | m.waca | di.waca | waca.an |
| | **Sastra.org (*Krama*)** | waos | maos | diwaos | waosan |
| | **Subword** | waos | m.waos | di.waos | waos.an |
| **Forms** | | root | active voice | passive voice | noun |
| **English translation** | | read | to read | to read | reading(s) |

## 2.2. Tesseract Optical Character Recognition (OCR)

Tesseract[1] is an OCR engine originally developed by Hewlett-Packard (HP) Co. Tesseract engine was released as an open source in 2005 and developed by Google from 2006 until November 2018. Most image files could be used as an input for this engine which translates to a text document containing characters recognized from the image. As of Tesseract version 4.0.0, the engine supports 121 languages including Javanese (tesseract-ocr, 2021). Although supporting most languages written using Latin alphabets, the Tesseract engine also supports several other scripts (tesseract-ocr, 2021). However, Javanese traditional script, which is also known as Carakan, is not included as one of the scripts supported by Tesseract OCR.

In this research, the Tesseract engine will be used to create a noisy (OCR) test set. The OCR test set is related to the research question (4) we would like to answer by analysing how models generalize over this type of documents. This type of document will be discussed further in Section 3.1.4.

[1] https://github.com/tesseract-ocr/tesseract

*Table 3 Original, SynOCR, OCR lines taken from historical datasets*

| | |
|---|---|
| Bab lêsing sidane priye ? sarta gone nyinau pranatan anyar bab barang kuna sing ta wènèhke kowe priye , wis diinggalke | ***Original (Manually transcribed)*** |
| Bab lêsing **sid5ane** priye ? sarta **kgone** nyinau pranatan anyar bab **barag** kuna sing ta wènèhke kowe priye , **wsi** diinggalke | ***SynOCR*** |
| **adab lening niaane nrêdesarto gonne niinaoe** pranatan **anjar** bab barang **koenn** sing ta wenehke kowe **nrije,win €4 ingenike** | ***Tesseract OCR*** |

## 2.3. Data Corruption

One of our research questions is whether models are able to generalize over noisy datasets, either transcription error or OCR failure, often found in Javanese historical documents. Since most historical documents were written using Javanese scripts, Optical Character Recognition (OCR) method was not possible using the Tesseract engine. Instead, the synthetic corruption generation method (SynOCR) is used to simulate corrupted data obtained from OCR. März et al. (2021) use this method to make manually transcribed datasets more similar to OCR datasets. Table 3 shows the comparison of the manually transcribed, SynOCR, and Tesseract OCR from our historical datasets. Below is our detailed method of corruption (März et al., 2021):

- **Insertion**. In this corruption method, a random character will be inserted randomly between a certain word. Characters inserted range from standard Latin alphabets without diacritics, numbers, or special characters (dot, comma, semicolon, etc.). We decided to include numbers and special characters which are not included by März et al. (2021) to make SynOCR as close as OCR datasets as possible. As observed in Table 3, the OCR result from Tesseract also includes numbers and special characters.
- **Deletion**. For this method, a random character is removed from the selected word.
- **Transpose**. This corruption method switches a character in the word with another character before or after it.

Referencing from previous research, our datasets are randomly corrupted 20% of the data by incorporating insertion, deletion, or transpose method (März et al., 2021). Our datasets are corrupted by first splitting the dataset into lists of five tokens. Between these five tokens, one word would be randomly chosen to be corrupted using one of the three corruption methods mentioned. To keep our consistency on corrupting the dataset, the corruption method chosen is also randomized.

## 2.4. Domain Adaptation

In NLP, domain refers to a coherent type of corpus which could be detailed as the relation in topics, styles, genres, or linguistic registers (Ramponi & Plank, 2020). In this research, the domains are divided into historical and modern datasets. For our historical Javanese datasets, we will be using documents collected from sastra.org digital library. The datasets will be referred to as Sastra datasets

in later sections. The modern dataset consists of documents collected from Wikipedia. In the following sections, this dataset will be referred to as Wikipedia dataset. The datasets will be further detailed in Section 3.1.

Domain adaptation is a particular type of transfer learning where there is a lack of labelled data on the target domain. The task in this field of study is to adapt the source domain into the target domain (Ramponi & Plank, 2020). In the case of this research, the source domain would be the Wikipedia dataset which has been used in previous research. The target domain for this research would be the Sastra datasets which have yet to be explored.

Wikipedia has become a common source of datasets. This means that the dataset is proved reliable and could become the reference for further research. Since the documents in the Wikipedia dataset are written after the 2000s, we are interested in knowing whether the models trained using Wikipedia could generalize over historical datasets or not. Another point of interest that could be studied in this research is whether word representations have enough robustness to represent words in the Sastra datasets.

Using a domain adaptation task, testing models over different domains, we will try to answer research questions (1), (2), and (3). How word representations are composed and how they generalize over different domains will be explored in this research.

The domain adaptation task could also be used to answer the research question (4), namely how models generalize over noisy input. In Table 3, it has been demonstrated how datasets compare from one to another. Using models trained over simulated noisy datasets (SynOCR), we aim to train the models to generalize over corrupted historical documents. A higher rate of generalization over corrupted documents indicates higher models' robustness over different test sets.

## 2.5. Word Representations

In recent natural language processing studies, words are represented as vectors. This method significantly simplifies and improves NLP applications (Mikolov, Chen, et al., 2013a). Through their research, they created a model to effectively represent words on large data sets. Mikolov, Chen, et al. (2013a) developed their model using continuous bag-of-word and skip-gram. The continuous bag-of-word model is a representation model where a word is represented based on its frequency of appearance in a text while continuous skip-gram is a word representation where a word is represented based on its neighbouring words.

Mikolov, Chen, et al. (2013a) model maps each word into a single, fixed dimensional size (e.g., d=300) which captures the word meaning. In vector space, similar words are placed closer compared to words that are very different or close to no relation at all. For example, the distance between vector("cat") and vector("dog) is smaller compared to vector("cat") and vector("fork"). Another example, the representation also captures an analogous relationship between words: Representation of

the word “Rome” equals to vector(“Paris”) – vector(“France”) + vector(“Italy”) (Mikolov, Chen, et al., 2013).

Bojanowski et al. (2017) later used this model (Mikolov, Chen, Corrado, & Dean, 2013a) as a baseline to compare the performance of word and subword vector representation. Bojanowski et al. (2017) developed this model using the character n-gram which segments a word based on *n* number of characters (see Section 2.6.4). The model combines character n-gram segmentation and continuous skip-gram used by Mikolov, Chen, et al. (2013a). This model is able to represent suffixes and prefixes as well as separate compound words into their morphemes which are found in morphologically rich languages. To compute word representation from its subword, Bojanowski et al. (2017) used addition as their composition function (Section 2.7).

Word vector representations or word embeddings are used to optimize various tasks in NLP. Various NLP tasks in previous works are summarized in Table 4. Word similarity and machine translation are supervised tasks, while language modelling is part of unsupervised tasks. Because of the lack of annotated data in Javanese, we will use language modelling (Section 2.8) to analyse the performance of our subword representation.

*Table 4 Previous work on word representations (Table taken from Vania & Lopez, 2017).*

| Models | Word/Subword Unit | Composition Function | Task |
|---|---|---|---|
| **(Mikolov, Chen, Corrado, & Dean, 2013a)** | Word | None | Language modelling |
| **(Sennrich, Haddow, & Birch, 2016)** | Morphs (BPE) | None | Machine translation |
| **(Bojanowski, Grave, Joulin, & Mikolov, 2017)** | Character n-gram | Addition | Language modelling |
| **(Labeau & Allauzen, 2017)** | Word, character, character n-gram, lemma + tags | Bi-LSTM, Char-CNN | Language modelling |
| **(Vania & Lopez, 2017)** | Character, character 3-gram, BPE, Morfessor | Addition, Bi-LSTM, CNN | Language modelling |
| **(Gerz, et al., 2018)** | Character | None | Language modelling |
| **(Kudo, 2018)** | BPE, subword regularization | None | Machine translation |
| **(Zhu, et al., 2019a)** | Morfessor, BPE, character n-gram | Addition | Fine-grained entity typing, morphological tagging, named entity recognition |
| **(Zhu, Vulić, & Korhonen, 2019b)** | BPE, Morfessor, CHIPMUNK | Addition, self-attention | Word similarity, dependency parsing, fine-grained entity typing |
| **(Provilkov, Emelianenko, & Voita, 2020)** | BPE, subword regularization, BPE-dropout | None | Machine translation |

## 2.6. Word Segmentation Methods

In this section, we present word segmentation methods that will be used in this research. There are two types of segmentation in this Section: Morph-based segmentations, which consist of Morfessor, Byte-Pair Encoding (BPE), and BPE-dropout; and character-based segmentation which consists of n-gram segmentation and character segmentation. Different segmentations affect the perplexity result of the language model depending on the language (Vania & Lopez, 2017; Labeau & Allauzen, 2017). For example, in research by Vania & Lopez (2017), character 3-gram segmentation works better for Finnish and Turkish with different composition functions. Meanwhile, character segmentation works better for Japanese, another agglutinative language used in the experiment (Vania & Lopez, 2017).

### 2.6.1. Morfessor

Morfessor is a model for word segmentation. Morfessor breaks down a *compound* (word) into *constructions* (morphs) (Smit et al., 2014). Morfessor baseline initially only has an unsupervised method where it breaks down a word into its subwords using minimum description length (MDL) principle; a principle where a simpler representation is preferred than literal description of data (Grunwald, 2004). Smit et al. (2014) improved the initial version of Morfessor into Morfessor 2.0 which includes semi-supervised extensions with only a small amount of annotated data for it to better understand the correct segmentation. Figure 1 shows a comparison between the unsupervised and semi-supervised methods of Morfessor 2.0 using Finnish.

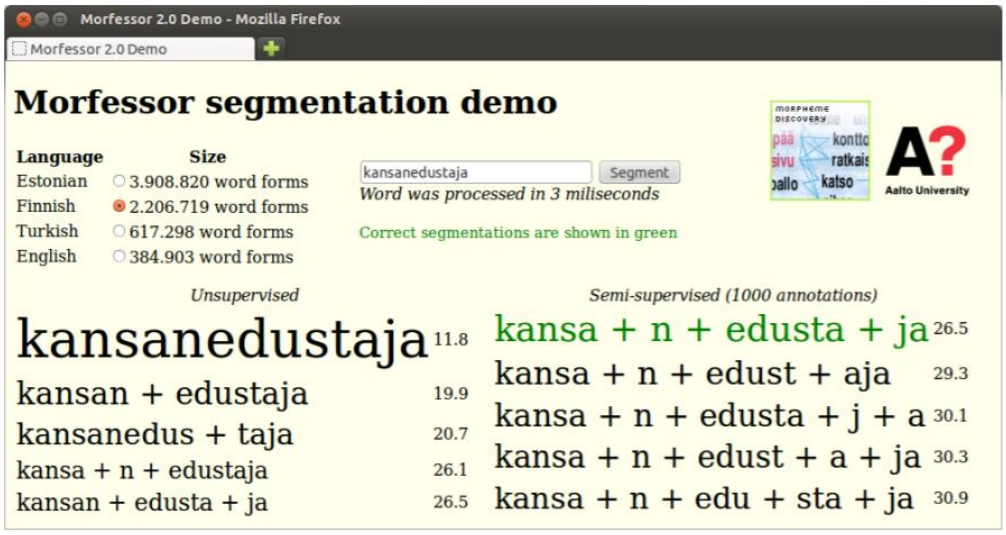



*Figure 1 Morfessor 2.0 segmentation (Figure taken from Smit et al., 2014)*

### 2.6.2. Byte-Pair Encoding

Byte-Pair Encoding (BPE) was initially proposed as a method for word compression (Gage, 1994). BPE works by replacing frequent sub-sequences of data with another representation. Figure 2 shows how the concept of BPE works in implementation. Given a sequence of data "ABABCABCD", "X" is used to represent "AB"; a frequent sequence that can be found in the initial data. Translation of "X" to "AB" is stored in a table where it will be used to return the compressed sequence into the initial sequence in decompressing process.

```
Original input data string:   ABABCABCD
Change pair AB to unused X:   XXCXCD
Change pair XC to unused Y:   XYYD
```

*Figure 2 How BPE works (Figure taken from Gage, 1994)*

Referring to the concept of BPE by Gage (1994), Sennrich et al. (2016) implemented BPE to segment a word into its subwords. Research by Sennrich et al. (2016) replaces common a sequence of characters ('a', 'b') with an unused representation of ('ab'). The final vocabulary size is equal to the size of the initial vocabulary plus the number of merge operations which serves as a hyperparameter of the BPE algorithm.

Using the example provided by Sennrich et al. (2016), dictionary {'low', 'lowest', 'newest', 'wider'} has frequent subwords {'low', 'est'}. Thus, the out-of-vocabulary (OOV) word of 'lower' will be represented as {'low', 'er'} (Sennrich et al., 2016). Using their translation task, this method successfully translated English words into German and Russian words which have different morphologies. Sennrich et al. (2016) show that BPE is effective for handling rare words in neural machine translation. This property of effectively handling rare words is beneficial in our case of *critical under-resourced* language and noisy dataset (Table 3).

### 2.6.3. BPE-dropout

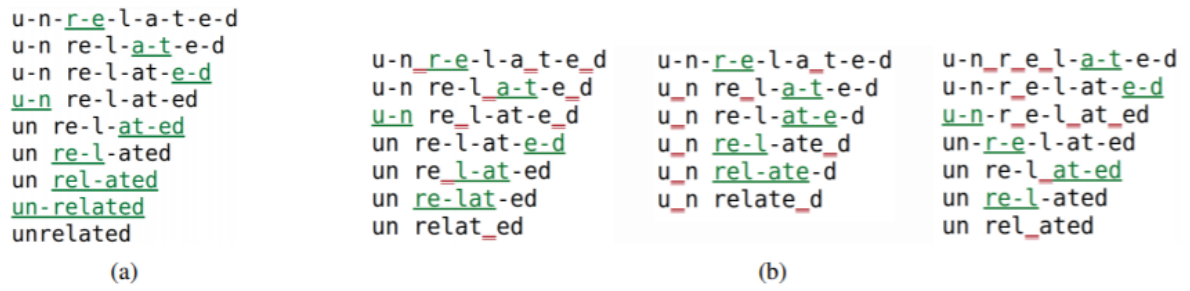



*Figure 3 BPE (a) and BPE-dropout (b) merges comparison. Red color in (b) indicates dropped merges while hyphens indicate possible merges (Figure taken from Provilkov, Emelianenko, & Voita, 2020).*

BPE-dropout is a further development of BPE where a random merge from a list of possible merges in BPE is deliberately dropped with a certain probability. Whenever a merge is dropped, words are segmented into different subwords (Provilkov et al., 2020). A comparison of BPE and BPE-

dropout can be seen in Figure 3 where BPE-dropout produces more subword variations compared to BPE.

Using neural language translation task, Provilkov et al. (2020) show that BPE-dropout performs better compared to BPE. Figure 4 shows that BPE-dropout can find closest vector representations of correct spellings as opposed to BPE that finds unrelated subwords through vector representations. Because it produces more variations, this method is more robust to corrupted or misspelled inputs (Provilkov et al., 2020) as found in our datasets.

withdra

| BPE | BPE-dropout |
|---|---|
| aimed | withd |
| molecules | withdrawal |
| aromatic | withdraw |
| specialties | withdrawn |
| publishers | withdrew |
| chain | withdrawals |
| americ | withdrawing |
| chron | dra |
| eager | retire |
| ighty | reti |

resul

| BPE | BPE-dropout |
|---|---|
| undert | result |
| checkl | results |
| maastr | resulting |
| & | resulted |
| unisp | ults |
| phili | res |
| ç | resultant |
| preca | ult |
| prosecut | ul |
| tali | outcome |

meeting

| BPE | BPE-dropout |
|---|---|
| meetings | meetings |
| meet | meet |
| session | eting |
| conference | me |
| met | etings |
| workshop | met |
| meets | meets |
| sessions | session |
| convened | et |
| reunion | conference |

olec

| BPE | BPE-dropout |
|---|---|
| olecular | molec |
| molecules | olecular |
| ljubl | molecule |
| zona | molecular |
| choler | molecules |
| oler | aec |
| ospheric | oler |
| olar | tolu |
| elic | omet |
| ochlor | olip |

comptroll

| BPE | BPE-dropout |
|---|---|
| icial | comptrollership |
| supervis | comptroller |
| & | troll |
| subcomm | controll |
| yugosl | contoller |
| trigg | controlled |
| sophistic | controllers |
| obstac | control |
| reag | contro |
| entals | controls |

*Figure 4 BPE and BPE-dropout nearest neighbour comparison of simulated corrupt data (Figure taken from Provilkov et al., 2020).*

### 2.6.4. Character-based segmentation

There are several ways to segment words based on characters. This method segments words regardless of morph/morphemes and frequency of subwords. Previous works on this method focus on character n-gram; segmentation of *n* amount of character (Labeau & Allauzen, 2017; Vania & Lopez, 2017; Bojanowski et al., 2017; Zhu, et al., 2019b) and character segmentation (Labeau & Allauzen, 2017; Vania & Lopez, 2017; Gerz, et al., 2018). In Table 5, different word segmentations are used to segment the Javanese word *ditumbasaken*.

*Table 5 Comparison between different word segmentations*

| Segmentation | σ(ditumbasaken) |
|---|---|
| **Morfessor** | di, tumbas, ake, n |
| **BPE** | di, t, um, ba, s, ak, en |
| **Character n-gram (n=3)** | dit, itu, tum, umb, mba, bas, asa, sak, ake, ken |
| **Character (Character unigram)** | d, i, t, u, m, b, a, s, a, k, e, n |

## 2.7. Composition Functions

Vania & Lopez (2017) compute word representation w given word ω as:

$$w = f\left(W_s, \sigma(\omega)\right) \qquad (1)$$

where $\sigma$ is a deterministic function that returns sequence of subword units taking ω as an input (Vania & Lopez, 2017). $W_s$ in Equation (1) is a matrix representation of $d$ dimensions for vocabulary

of subword units, while $f$ is a composition function to calculate w by taking $W_s$ and σ(ω) as parameter and returns the representation of word ω.

Composition function is a function that we use to combine subword embeddings into word-level embeddings before calculating its probability in language model. Given $\omega = sw_0, \dots, sw_n$ where $sw_i$ represents subword embeddings, we can calculate word representation *w* using:

- **Addition** (Bojanowski et al., 2017; Vania & Lopez, 2017; Zhu, et al., 2019a; Zhu, Vulić, & Korhonen, 2019b). This composition function constructs representation of word w by adding representation of its subwords:

$$\mathrm{w} = \sum_{i=0}^{n} sw_i \qquad (2)$$

- **Multiplication** (Zhu, Vulić, & Korhonen, 2019b). This composition function represents word w by multiply all its subword representations:

$$\mathrm{w} = \prod_{i=0}^{n} sw_i \qquad (3)$$

- **Bidirectional Long-Short Term Memory (Bi-LSTM)**. Just like LSTM, Bi-LSTM also takes an input of sequence. Instead of one-way LSTM (Hochreiter & Schmidhuber, 1997), Bi-LSTM consists of LSTM going forward and another LSTM going backward over the sequence (Graves, Fernández, & Schmidhuber, 2005). In the case of character-level Bi-LSTM, the input is a sequence of characters (Vania & Lopez, 2017) instead of sequence of words. Vania & Lopez (2017) implemented character-level Bi-LSTM proposed by Ling et al. (2015) to compute word representations. Character-based Bi-LSTM implemented by Vania & Lopez (2017) produces word representation by modelling "complex non-local dependencies between characters in a given word" (Vania & Lopez, 2017).

  Figure 5 illustrates architecture of CharBi-LSTM used by Ling et al. (2015) and Vania & Lopez (2017). CharBi-LSTM started by listing a set of characters *C* used in a language and word $\omega = c_0, \dots, c_n$. Each character in word ω then represented using one-hot encoding before mapped into each character embeddings (Vania & Lopez, 2017). Because it takes a sequence as an input, LSTM runs based on *time step*. On each step *i*, a character in the sequence is fed into each forward and backward LSTM with forward LSTM going over the sequence and backward LSTM going in reverse sequence. LSTM computes hidden state $h_i$ of each character using Equation (4) considering the hidden state of previous step $h_{i-1}$ (Vania & Lopez, 2017).

$$h_i = LSTM(c_i, h_{i-1}) \qquad (4)$$

  Word representation is computed by combining both hidden states into heir parameter matrices that are learned during training. $W_f$ is a matrix representation of word ω generated from forward LSTM, while $W_b$ is matrix representation learned from backward LSTM. To calculate word representation $w_t$ from word ω, Vania & Lopez (2017) concatenate final

hidden states of forward and backward LSTM with their respective representation matrices. $b$ in Equation (5) indicates bias parameter for LSTM.

$$w_t = \boldsymbol{W_f} \cdot h_n^f + \boldsymbol{W_b} \cdot h_0^b + b \qquad (5)$$

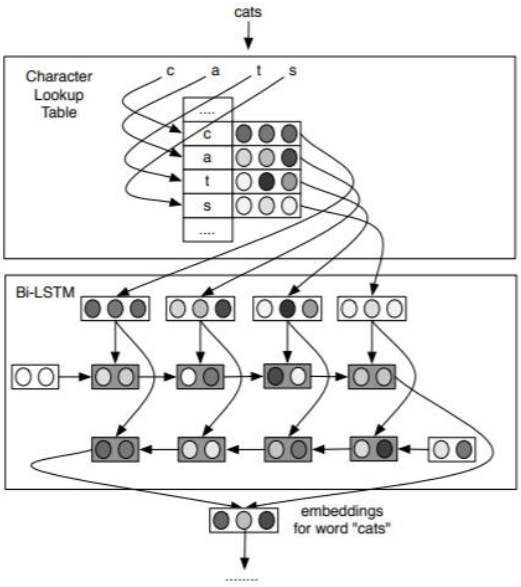



*Figure 5 Illustration fo CharBi-LSTM model (Figure taken from Ling et al., 2015)*

- **Convolutional Neural Network (CNN)** (Labeau & Allauzen, 2017; Vania & Lopez, 2017). Kim et al. (2016) proposed a model to compute word representations from their characters using CNN. Similar to character-level Bi-LSTM, set of characters *C* in a language is used in character-level CNN. Given a word $\omega = c_0, \dots, c_n$, character embedding matrix of $\boldsymbol{C_\omega} \in \mathbb{R}^{d \times n}$ is built where embedding of each character $c_i$ occupies a column (Kim et al., 2016; Vania & Lopez, 2017). In a research by Vania & Lopez (2017), zero padding between $\boldsymbol{C_\omega}$ and filter $\boldsymbol{F} \in \mathbb{R}^{d \times m}$ of width *m* was applied in addition to bias *b* and non-linearity function $tanh$ to obtain feature map $\boldsymbol{f_\omega} \in \mathbb{R}^{n-m+1}$ (Kim et al., 2016). *j*-th element of $\boldsymbol{f_\omega}$ is calculated using Equation (6):

$$\boldsymbol{f_\omega}[j] = tanh(\langle \boldsymbol{C_\omega}[:, j: j+n-1], \boldsymbol{F} \rangle + b) \qquad (6)$$

where $\boldsymbol{C_\omega}[:, j: j+n-1]$ is the *j*-to-*(j+n-1)*-th column of $\boldsymbol{C_\omega}$ and $\langle \boldsymbol{A}, \boldsymbol{B} \rangle$ is Frobenius inner product (Kim et al., 2016). Max-over-time $y_\omega$ is then calculated as a feature corresponding to filter $\boldsymbol{F}$ (Kim et al., 2016).

$$y_\omega = \max_j \boldsymbol{f_\omega}[j] \qquad (7)$$

According to Kim et al. (2016), word embedding could be directly replaced with $y_\omega$. However, Kim et al. (2016) choose to run $y_\omega$ through a highway network which improves its performance. Highway network uses concatenation of max-over-time of each filter $y = [y_0, \dots, y_p]$ where *p* is number of filters applied (Vania & Lopez, 2017). As formulated by Kim et al. (2016) and Clara & Lopez (2017), highway network does the following:

$$\boldsymbol{z} = t \odot g(W_F \cdot y + b_F) + (1 - t) \odot y \quad (8)$$

where g is a non-linearity function, $t = \sigma(W_T \cdot \mathrm{y} + b_T)$ is a transform gate, and $(1 - t)$ is a carry gate; with $W_F$ as weight matrix and $b_F$ as bias on each filter (Kim et al., 2016; Vania & Lopez, 2017) Figure 6 illustrates the structure of character-level CNN used by Kim et al. (2016) and Vania & Lopez (2017).

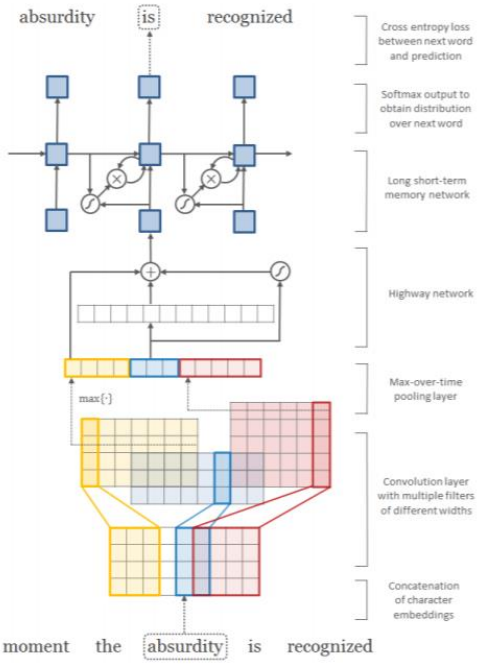



*Figure 6 Character-level CNN (Figure taken from Kim et al. (2016))*

What word embeddings are has been explained in Section 2.5. In Section 2.6, methods to form subwords are described. Equation (1) mathematically explains how word embeddings are composed from subwords using different composition functions (Section 2.7). These word embeddings will be an input to our language models which will be further explained in Section 2.8.

In this research, we will be using different combinations of subwords and composition functions to compare which combination works best for different domains along with word-based representations as a baseline. We will be exploring three segmentation methods in this research, namely BPE (Section 2.6.2), BPE-Dropout (Sections 2.6.3), and Character-3gram (Section 2.6.4) with Addition and BiLSTM composition functions. Other segmentations, such as Morfessor and Character-unigram along with Multiplication and CNN composition methods will be explored in future research.

## 2.8. Language Modelling

Language modelling has been studied in the history of natural language processing for a long time. A language model is a probabilistic technique to determine probability of a sequence of words

occurring in a sentence. Language model is used in various NLP tasks such as machine translation or grammatical and fluency analysis of a text.

Given a sequence of words $s = \omega_0, \ldots, \omega_n$, a language model computes probability of a sequence as:

$$P(s) = \prod_{i=0}^{n} P(\omega_i | \omega_0, \ldots, \omega_{n-1}) \qquad (9)$$

In Equation (9), probability of a plausible sequence (of words) is computed based on previous words in the sentence. We next will introduce several language modelling methods used in this research.

### 2.8.1. Recurrent Neural Network (RNN)

In 2010, Mikolov et al. (2010) implemented RNN for language modelling. This method allows the neural network to take the context from previous units and use it as a parameter to compute the next possible unit. The advantage of using RNN over previous non-neural method like Kneser-Ney 5-gram is its ability to learn information without size limitation of the context (Mikolov et al., 2010).

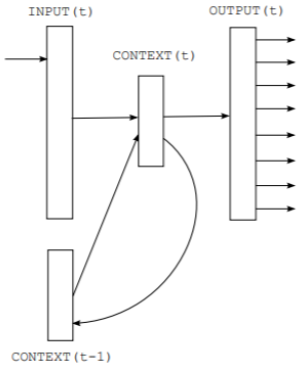



*Figure 7 Recurrent Neural Network (Figure taken from Mikolov et al., 2010)*

### 2.8.2. Long-Short Term Memory (LSTM)

Vanilla RNN is prone to fall into vanishing or exploding gradient problems; problems where the gradient keeps falling to zero or rising to infinity. Because of the side effect, LSTM is preferred over vanilla RNN. Vania & Lopez (2017) implemented LSTM variant of RNN for their language model. This model solves the vanishing gradient problem because LSTM has the property to selectively "forget" and "remember" old information in the sequence. Figure 8 shows how LSTM variant of RNN is used in language modelling using softmax function.

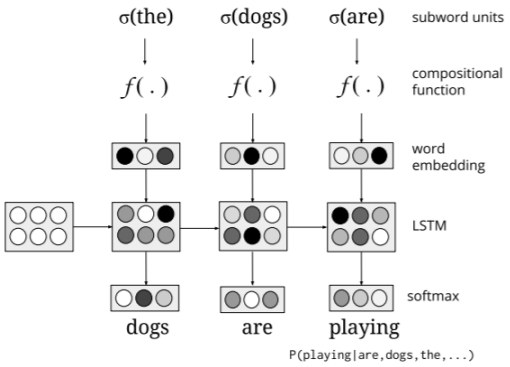



*Figure 8 LSTM Language Modelling (Figure taken from Clara & Lopez, 2017)*

RNN runs on time steps. This process is done recursively, meaning RNN can take unlimited amount context given a sequence $s = \omega_0, \ldots, \omega_n$. Similar to LSTM in general, RNN takes a sequence as an input. Vania & Lopez (2017) used LSTM variant of RNN where on each step *t*, the model takes an input of embeddings $w_t$ which is a representation of word $\omega_t$. Representation $w_t$ is then passed into Equation (10) where the new hidden state $h_t$ is calculated based on previous state $h_{t-1}$:

$$h_t = LSTM(w_t, h_{t-1}) \qquad (10)$$

Word $\omega_{n+1}$ is predicted using $\widehat{w}_{t+1}$ which is the most likely word given a probability distribution over vocabulary (Vania & Lopez, 2017).

$$\widehat{w}_{t+1} = softmax(V^T \cdot h_t) \qquad (11)$$

This research will be using the RNN-LSTM language model. The design is implemented using Tensorflow libraries (Abadi, et al., 2015). The language models then will be evaluated using metrics described in Section 3.3.

### 2.8.3. Language Model Design

In this research, Vania & Lopez (2017) language model design[2] is used. The design uses the LSTM variant of RNN which prevents the gradients from falling to zero or rising to infinity. General language model design takes word embeddings produced from the embeddings layer and calculates the probability of the next word. In the RNN-LSTM case, the language model takes the hidden state produced from the previous step and weight/embedding of the current word to predict the next word. RNN only takes previous words as much as specified time step *t* (see Section 4.1) to prevent overfitting.

The model takes word embeddings as input and calculate them using LSTM. In addition to word embeddings, the hidden state from the previous calculation also becomes an input to give context to the current word (refer to Equation 10). Representation of the embedding units are adjusted during the training of the model by optimizing gradient descent. The concept of gradient descent is tweaking parameters during the training until they fitted into the context of the target word $\widehat{w}_{t+1}$.

[2] https://github.com/claravania/subword-lstm-lm

The language model design follows a closed vocabulary, where unknown words not encountered during training will be assigned to an unknown token for the baseline/word model. This will be a problem during testing since different words with different contexts are assigned into unknown token embeddings. For example, if the word **"fork"** and **"rabbit"** are not found during the training phase, they will have the same representation. Such a case is the disadvantage of word-based representation.

During the training process, a word with a single occurrence will be stochastically replaced with the unknown token with 0.5 probability (Vania & Lopez, 2017). Using this way, the representation of unknown tokens could be calculated by the model since unknown tokens will never be encountered in actual documents. After vocabulary is built, the model would then compose word/subword embeddings using the model explained previously.

Different from word-based embeddings, subword-based models split the word into its subwords and create new representation using composition methods described in Section 2.7. This is the reason why subword-based embeddings are more robust compared to word-based embeddings. However, since the design is unable to predict words outside of a limited output dictionary, the target word $\widehat{w}_{t+1}$ will be assigned as unknown token. Even though the prediction falls closer to an unknown token, the model is able to recover from the confused state by using hidden state and input word embeddings calculated from subwords representations computed using LSTM.

In their research, Vania & Lopez (2017) compared four different subword units: Morfessor, BPE, character, and character 3-gram (Section 2.6.1-2.6.4); combined with three different composition functions: addition, Bi-LSTM, and CNN (Section 2.7). They measured the performance of subword embeddings using perplexity (Section 3.2), which is an evaluation method for language models. However, Vania & Lopez (2017) did not mention which combination works best for agglutinative languages in particular Javanese, which is the focus of this research.

### 2.8.4. Open Dictionary Language Model

The language model discussed in Section 2.8.3 can only predict words from a finite output dictionary. The limitation restricts so that it could not form a new word from known subwords. A language model design proposed by Kim G. (2019) could overcome this problem by composing a new word from subwords. The proposed design aims to improve query auto-completion which relies on character-level language models.

The language model takes subwords as an input: BPE (Sennrich et al., 2016), Subword Regularization (Kudo, 2018), and character as a baseline. Using Equation (9), subword $\hat{d}$ is used instead of word ω. The model predicts the next subword $\hat{d}_{t+1}$ given subword representation $d$ using:

$$\hat{d} = argmax \log P(d) \qquad (10)$$

The model then concatenates predicted subwords $\hat{d}$ to return them into proper words $\hat{w}$ :

$$\hat{w} = concat(\hat{d}) \qquad (11)$$

Character-level language models tend to be slower and inaccurate when they receive a long sequence of queries (Kim G. , 2019). The design could be improved by implementing BPE segmentation (Sennrich et al., 2016) and subword regularization (Kudo, 2018). Instead of calculating character representations, the models could predict subwords by taking the previous context and subword representations. Since the granularity of subwords is higher than characters, the calculation process could be shortened.

The open-dictionary language model proves to be more accurate and has faster processing speed compared to character-level language models (Kim G. , 2019). However, the design focuses on taking inputs from search queries. Hence, there are low-level semantic connections between input words. Our research focuses more on long sequences of words that have high-level semantic connections. Thus, the design is not directly transferable for our research. Although the possibility of incorporating the design into our language models is an interesting direction to be explored in the future research.

# 3. Methodology

This chapter is divided into three sections: Datasets, Preprocessing, and Evaluation. In this first section of this chapter, we will be discussing the datasets used in this research. The datasets consist of historical (Sastra) and modern (Wikipedia) datasets. The detailed explanation about the datasets will also be presented in Section 3.1.1 to 3.1.4, followed by preliminary analysis of the dataset in Section 3.1.5.

In the following section, we will discuss preprocessing made to the datasets. This preprocessing is added to make our datasets in a similar format with our reference (Ling, et al., 2015; Vania & Lopez, 2017). Detailed steps of how preprocessing is done will be explained in Section 3.2.

In the final section of this chapter, we propose several metrics that could be used to analyse our experiment results. In this research, we will be using perplexity score as our main evaluation metric. How perplexity scores are calculated will be detailed in Section 3.3.1.

## 3.1. Datasets

Our datasets consist of articles from Wikipedia and Sastra dataset. Wikipedia is chosen as our modern dataset because it was recently written (after the 2000s), while Sastra datasets serve as historical datasets. Two datasets are used in this research to answer our research questions (Section 1.2) which includes how historical and modern Javanese interact and how much different they are.

Sastra dataset is an online library created by Yayasan Sastra Lestari, a non-profit organization in Indonesia focusing on transcribing historical Javanese publications and literature. The publications which are originally written using Carakan (traditional Javanese script) are digitized and hand-transcribed by Javanese experts and philologists into Latin alphabet. Transcribers are proficient speakers of (new) Javanese so that the resulting documents (refer to Figure 10) are reliable to be used as manually transcribed documents and constitute a valuable resource for NLP research focusing on historical Javanese.

Documents in the Sastra dataset are sourced from various authors. The documents are originally literature collected by museums and from private collections. After the documents are digitized and transcribed, they are returned to their respective owners (Yayasan Sastra Lestari, 2021). Most private collections are preserved by descendants of nobles residing in Central Java. Following our discussion with John Paterson (john.paterson@sastra.org), Sastra dataset licensing policy follows Creative Commons Attribution-NonCommercial 4.0 International (CC BY-NC 4.0). This licensing policy allows the data to be shared and adapted with attribution to the licensor for non-commercial purposes. The usage of this licensing policy means that the dataset is open data and suitable for research purposes.

Yayasan Sastra Lestari grouped the digitized documents under different categories. There are five categories of historical datasets which are further described in Table 6. The crawled documents

are tagged by the Sastra Lestari team which later becomes our basis to categorize the documents. Table 6 also details how many documents are crawled from the Sastra website along with which categories it is under. As observed in Table 6, documents are dated from the mid-19th century 20th century with some outliers from the late 18th century.

Categories in Table 6 are what is called domains in this research. The focus of this research is to observe how different models generalizes over different domains. In this case, models are trained to adapt to different domains (categories). Three different domains are used in this research, namely: Religions and Beliefs (RB), Archive and History (AH), and Wikipedia (WIKI). These domains are chosen to compare how models trained on historical (RB and AH) and modern (WIKI) datasets interact with each other. Further information about domain adaptation is discussed in Section 2.4.

*Table 6 Sastra dataset properties*

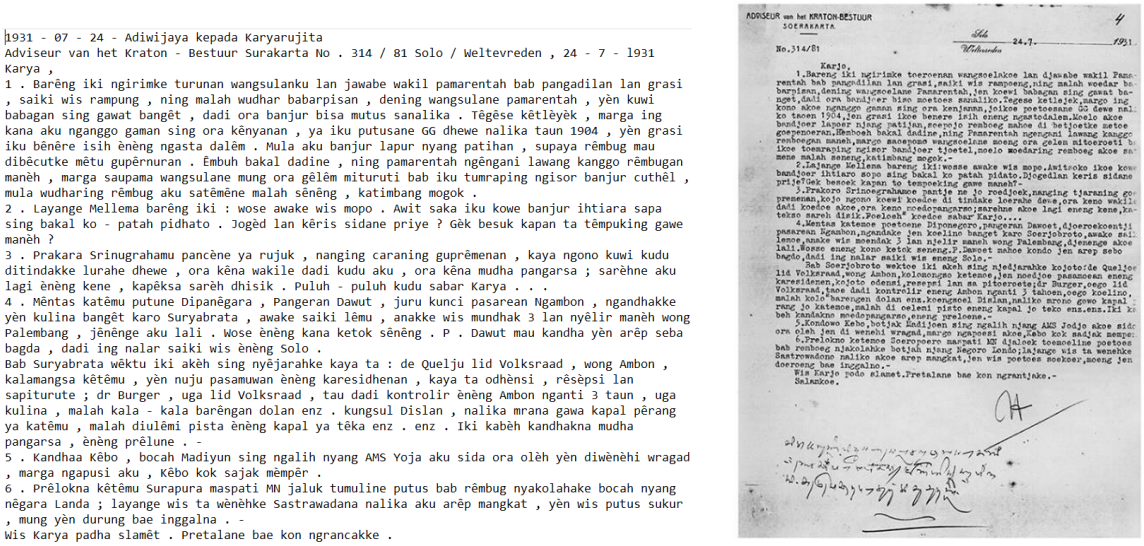
| Category | Sastra Dataset | | | | | Wikipedia (WIKI) |
|---|---|---|---|---|---|---|
| | Religions and Beliefs (RB) | Archives and History (AH) | Language and Cultures (LC) | Stories and Chronicles (SC) | News, Magazines, and Journals (NMJ) | |
| **Average year** | 1917 | 1914 | 1921 | 1925 | 1927 | - |
| **Year range** | 1855 - 1960 | 1789 - 1997 | 1841 - 1995 | 1788 - 1985 | 1858 - 1955 | - |
| **# Crawled Documents** | 229 | 797 | 541 | 556 | 685 | 25,575 |
| **# Words** | 1,545,514 | 2,024,619 | 2,763,568 | 7,809,599 | 4,345,203 | 8,423,759 |

Dataset for our modern documents is collected from Javanese Wikipedia. Wikipedia is chosen because it is easily accessible and contains a large resource. A large percentage of Wikipedia articles are written using *Ngoko* style (Section 2.1) with a small portion written using *Krama* style (Section 2.1). This is a contrast to historical documents which are mostly written using Krama style. Errington (1992) further states that Javanese culture and language are well-preserved within nobility lineage which in this case also includes royal descendants from the previously mentioned principalities.


*Figure 9 Year distribution of historical datasets "Religion and Beliefs" (RB) and "Archive and History" (AH)*

Below, we will describe the datasets used in this research in detail, namely: RB, AH, and WIKI.

### 3.1.1. Archive and History (AH)



```
1931 - 07 - 24 - Adiwijaya kepada Karyarujita
Adviseur van het Kraton - Bestuur Surakarta No . 314 / 81 Solo / Weltevreden , 24 - 7 - 1931
Karya ,
1 . Barêng iki ngirimke turunan wangsulanku lan jawabe wakil pamarentah bab pangadilan lan grasi
, saiki wis rampung , ning malah wudhar babarpisan , dening wangsulane pamarentah , yèn kuwi
babagan sing gawat bangêt , dadi ora banjur bisa mutus sanalika . Têgêse kêtlèyèk , marga ing
kana aku nganggo gaman sing ora kênyanan , ya iku putusane GG dhewe nalika taun 1904 , yèn grasi
iku bênêre isih ènèng ngasta dalêm . Mula aku banjur lapur nyang patihan , supaya rêmbug mau
dibêcutke mêtu gupêrnuran . Êmbuh bakal dadine , ning pamarentah ngêngani lawang kanggo rêmbugan
manèh , marga saupama wangsulane mung ora gêlêm mituruti bab iku tumraping ngisor banjur cuthêl ,
mula wudharing rêmbug aku satêmêne malah sênêng , katimbang mogok .
2 . Layange Mellema barêng iki : wose awake wis mopo . Awit saka iku kowe banjur ihtiara sapa
sing bakal ko - patah pidhato . Jogèd lan kêris sidane priye ? Gèk besuk kapan ta têmpuking gawe
manèh ?
3 . Prakara Srinugrahamu pancène ya rujuk , nanging caraning guprêmenan , kaya ngono kuwi kudu
ditindakke lurahe dhewe , ora kêna wakile dadi kudu aku , ora kêna mudha pangarsa ; sarèhne aku
lagi ènèng kene , kapêksa sarèh dhisik . Puluh - puluh kudu sabar Karya . . .
4 . Mêntas katêmu putune Dipanêgara , Pangeran Dawut , juru kunci pasarean Ngambon , ngandhakke
yèn kulina bangêt karo Suryabrata , awake saiki lêmu , anakke wis mundhak 3 lan nyêlir manèh wong
Palembang , jênênge aku lali . Wose ènèng kana ketok sênêng . P . Dawut mau kandha yèn arêp seba
bagda , dadi ing nalar saiki wis ènèng Solo .
Bab Suryabrata wêktu iki akèh sing nyêjarahke kaya ta : de Quelju lid Volksraad , wong Ambon ,
kalamangsa kêtêmu , yèn nuju pasamuwan ènèng karesidhenan , kaya ta odhènsi , rêsèpsi lan
sapiturute ; dr Burger , uga lid Volksraad , tau dadi kontrolir ènèng Ambon nganti 3 taun , uga
kulina , malah kala - kala barêngan dolan enz . kungsul Dislan , nalika mrana gawa kapal pêrang
ya katêmu , malah diulêmi pista ènèng kapal ya têka enz . enz . Iki kabèh kandhakna mudha
pangarsa , ènèng prêlune . -
5 . Kandhaa Kêbo , bocah Madiyun sing ngalih nyang AMS Yoja aku sida ora olèh yèn diwènèhi wragad
, marga ngapusi aku , Kêbo kok sajak mèmpêr .
6 . Prêlokna kêtêmu Surapura maspati MN jaluk tumuline putus bab rêmbug nyakolahake bocah nyang
nêgara Landa ; layange wis ta wènèhke Sastrawadana nalika aku arêp mangkat , yèn wis putus sukur
, mung yèn durung bae inggalna . -
Wis Karya padha slamêt . Pretalane bae kon ngrancakke .
```

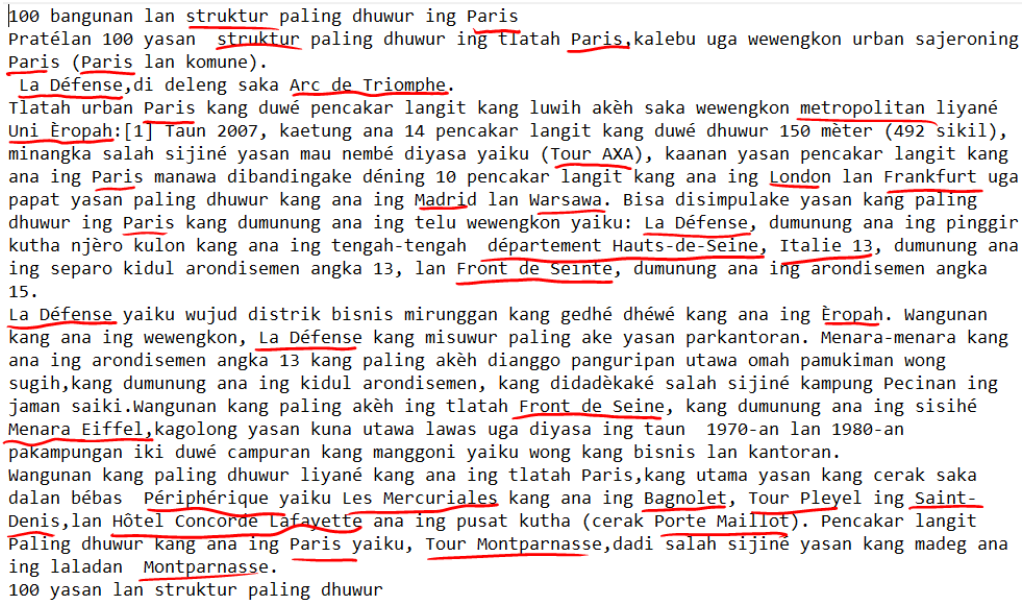
*Figure 10 (a) Left: Transliterated Sastra document (manually transcribed) (b) Right: Original document from AH category (Figure taken from Yayasan Sastra Lestari, 2021)*

AH category contains government archives, official letters, and a written law system between 1789 and 1956 in Java. Figure 9 (right) shows the year distribution of documents used in AH dataset. Most of these documents are written using Carakan (Javanese script) with a small amount of them written using Latin alphabets. Some of the documents contain loanwords that are mainly taken from Dutch (observed in Figure 10 (a)). Based on our crawl on June 27th, 2021, there are a total of 797 documents in this category (refer to Table 6).

### 3.1.2. Religion and Beliefs (RB)

RB category contains holy scriptures and some old Javanese beliefs system written between 1855 and 1962. Figure 9 (left) shows the year distribution of documents used in RB dataset. Holy scriptures in this category are divided into Javanese-translated Koran and some biblical texts from the book of Genesis. Documents in this category are written using Carakan (Javanese script) and contain loanwords from Old Arabic and biblical texts. Since most of the documents are written in form of poems instead of sentences, vertical bars ("|") are used as a separator. Based on our crawl on June 27th, 2021, there are a total of 229 documents in this category (refer to Table 6).

### 3.1.3. Wikipedia (WIKI)



100 bangunan lan struktur paling dhuwur ing Paris
Pratélan 100 yasan struktur paling dhuwur ing tlatah Paris,kalebu uga wewengkon urban sajeroning Paris (Paris lan komune).
La Défense,di deleng saka Arc de Triomphe.
Tlatah urban Paris kang duwé pencakar langit kang luwih akèh saka wewengkon metropolitan liyané Uni Èropah:[1] Taun 2007, kaetung ana 14 pencakar langit kang duwé dhuwur 150 mèter (492 sikil), minangka salah sijiné yasan mau nembé diyasa yaiku (Tour AXA), kaanan yasan pencakar langit kang ana ing Paris manawa dibandingake déning 10 pencakar langit kang ana ing London lan Frankfurt uga papat yasan paling dhuwur kang ana ing Madrid lan Warsawa. Bisa disimpulake yasan kang paling dhuwur ing Paris kang dumunung ana ing telu wewengkon yaiku: La Défense, dumunung ana ing pinggir kutha njèro kulon kang ana ing tengah-tengah département Hauts-de-Seine, Italie 13, dumunung ana ing separo kidul arondisemen angka 13, lan Front de Seinte, dumunung ana ing arondisemen angka 15.
La Défense yaiku wujud distrik bisnis mirunggan kang gedhé dhéwé kang ana ing Èropah. Wangunan kang ana ing wewengkon, La Défense kang misuwur paling ake yasan parkantoran. Menara-menara kang ana ing arondisemen angka 13 kang paling akèh dianggo panguripan utawa omah pamukiman wong sugih,kang dumunung ana ing kidul arondisemen, kang didadèkaké salah sijiné kampung Pecinan ing jaman saiki.Wangunan kang paling akèh ing tlatah Front de Seine, kang dumunung ana ing sisihé Menara Eiffel,kagolong yasan kuna utawa lawas uga diyasa ing taun 1970-an lan 1980-an pakampungan iki duwé campuran kang manggoni yaiku wong kang bisnis lan kantoran.
Wangunan kang paling dhuwur liyané kang ana ing tlatah Paris,kang utama yasan kang cerak saka dalan bébas Périphérique yaiku Les Mercuriales kang ana ing Bagnolet, Tour Pleyel ing Saint-Denis,lan Hôtel Concorde Lafayette ana ing pusat kutha (cerak Porte Maillot). Pencakar langit Paling dhuwur kang ana ing Paris yaiku, Tour Montparnasse,dadi salah sijiné yasan kang madeg ana ing laladan Montparnasse.
100 yasan lan struktur paling dhuwur

*Figure 11 Scraped Wikipedia article with red lines indicate foreign words in the document.*

Wikipedia documents are written after the 2000s. Since the documents were written after the national movement of Indonesia, they tend to receive influences from Indonesian. In addition, the modern dataset is also influenced by other foreign languages, such as English. Figure 11 shows one of our scraped Wikipedia documents titled "100 Highest Buildings and Structures in Paris" with loanwords indicated using a red underline. The document shows that foreign language has a higher influence on modern Javanese literature compared to historical documents (see Figure 10 (a)). The fact that historical documents have a lower rate of foreign words makes it a reliable source for the models to learn historical Javanese from.

We categorize Wikipedia documents as our modern documents. Different from historical documents which are written using *Krama* style, most Wikipedia documents are written using *Ngoko* style (see Section 2.1). Documents in Wikipedia are written by contributors with proficiency in Javanese. Based on our crawl on April 1st, 2021, we collected a total of 25,575 documents of this category.

### 3.1.4. OCR Documents

There were a limited number of documents that could be recognized using the Tesseract OCR (Section 2.2) engine, mainly because most historical documents were written in Javanese traditional script (Carakan) but only Latin alphabet is supported by Tesseract. To detail our OCR datasets, there are 22 documents from AH titled "Adiwijaya kepada Karyarujita", meaning: "Adiwijaya ('s letters) to Karyarujita". These documents were written using the Latin alphabet which means that Tesseract OCR (see Section 2.2) can be used to transcribe the documents. This document group only makes up 14 out of 538 documents included in the AH dataset (Table 8), with 11 documents in the training set,

two documents in the development set, and three documents in the test set. We will be using this dataset as a test set to evaluate models' performance on noisy documents (Section 4.4).

### 3.1.5. Datasets Similarity

Table 7 shows the similarity between our datasets. Datasets were compared using cosine similarity and vectorized using term frequency–inverse document frequency (tf-idf). Cosine similarity is a commonly used measure of similarity by comparing vector of word weights in a document, which in this case is the dot product of each document. The result from this cosine similarity would range from 0 to 1 which indicates how similar it is with 1 as exact copy and 0 as no shared dictionary.

$$\cos\theta = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \quad (12)$$

Equation (12) illustrates how cosine similarity is calculated. Cosine similarity is calculated by dividing sum of multiplication from weights of same terms from the document $A$ and $B$ with the normalized sum of term weights from the document A multiplied with normalized sum of term weights in the document $B$. In our case, term weight means tf-idf value which indicates the value of a term in a document in reference to the whole corpus.

Term weight, tf-idf, is used to calculate the value of a word in documents while observing how often the term appears in other documents. Tf value increases for every occurrence of a term in a document (Luhn, 1957) while idf balances it out by reducing the value if the term is frequently encountered in other documents (Jones, 1972). Using this method, tf-idf could better represent the significance of a word in a document while reducing the value of frequently used words that have lower importance such as stop words.

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \quad (13)$$

$$idf(t,D) = -\log\frac{1+|\{d \in D : t \in d\}|}{|D|} \quad (14)$$

$$tfidf(t,d,D) = tf(t,d) \cdot idf(t,D) \quad (15)$$

As illustrated in Equation (13), tf is calculated by dividing the frequency of a term found in the document $f_{t,d}$ with a total number of tokens $\sum_{t' \in d} f_{t',d}$ in the document. Idf is calculated using Equation (14) using negative logarithmic of the number of documents containing term $t$ divided by the total number of documents in the corpus. The numerator of the fraction is added by 1 to prevent logarithmic function resulting in $\log 0$ which translates to undefined number. The weight of the word is then calculated by multiplication of tf and idf value as illustrated in Equation (15).

Based on the calculation using Equation (12) and (15), AH and RB are more similar to one another than they are to modern datasets (WIKI). Similarities shown in Table 7, correlate with temporal proximity. Referring to Figure 9, AH and RB datasets were written between the same time period whereas the WIKI dataset was written after the year 2000.

*Table 7 Cosine similarity between datasets*

| | AH | RB | Wikipedia |
|---|---|---|---|
| **AH** | 1 | 0.813 | 0.664 |
| **RB** | 0.813 | 1 | 0.634 |
| **Wikipedia** | 0.664 | 0.634 | 1 |

## 3.2. Preprocessing

Scraped documents are preprocessed before they could be used to train (subword) language models. The first part of dataset preprocessing is filtering Wikipedia articles and using documents with a minimum of 100 words. The filtering process is not done to our historical datasets because all of them contains more than 100 words.

The second part of our preprocessing is by making our datasets comparable to previous research (Ling, et al., 2015; Vania & Lopez, 2017). Based on observation on the dataset used in previous research (Ling, et al., 2015), each document in the dataset is separated by a document separator token ("</doc>"). For modern and historical datasets links were removed and a document separator token is added after each document in the datasets. In addition, space is added after and before special characters (dots, comma, hyphens, etc.), which was also done in Ling et al. (2015).

Historical datasets further receive preprocessing to remove images, advertisements, page breaks, and page number identifiers that were not present in the original document and only present in transliterated documents. Additional preprocessing on the historical dataset is needed to make it similar to the Wikipedia dataset which does not contain said identifiers.

Table 8 describes the number of word tokens, types, and sentences in each document category. Two out of five categories described from the Sastra dataset are used in addition to the Wikipedia dataset because each category contains more tokens compared to datasets used in previous research (refer to Table 10 in Section 4.2). To make our datasets comparable, a maximum number of tokens is limited to 1.5M tokens before any second and third part of preprocessing. This means not all the documents collected from Sastra and Wikipedia described in Table 6 are directly used as datasets. Documents are shuffled and randomly selected from the list until 1.5M tokens are collected for the dataset.

*Table 8 Experiment dataset summary*

| | Archive and History (AH) | Religion and Beliefs (RB) | Wikipedia |
|---|---|---|---|
| **Token** | 1792245 | 1778925 | 2066563 |
| **Types** | 94650 | 86602 | 159257 |
| **Sentences** | 82309 | 81909 | 120661 |
| **Documents used in train-dev-test set** | 538 | 228 | 4598 |

## 3.3. Evaluation

In our evaluation, we aim to answer our research questions (Section 1.2) using the metrics described below. The first metric is perplexity (PPL) which is an unsupervised metric to measure how perplexed a language model (Section 2.8.3) is when exposed to a new sequence. The lower number produced from this metric indicates less perplexed a model is to a sequence (Eisenstein, 2018). The second metric to evaluate our model is using word error rate (WER) which measure how many errors the model generates when predicting words in a sequence.

### 3.3.1. Perplexity

There is no annotation for our datasets, thus supervised tasks such as word similarity or machine translation are impossible. For this research, the language modelling task is used and evaluated without relying on any annotation. The standard evaluation metric for the language model is called **perplexity (PPL)** (Eisenstein, 2018). Perplexity is calculated using average negative log probability $H$. Given $N$ is the length of the sentence:

$$PPL = 2^H \qquad (14)$$

where $H$:

$$H = \frac{1}{N}\sum_{i=0}^{N} -\log_2 P(w_i|w_0 \dots w_{i-1}) \qquad (15)$$

Equation (15) computes the average negative log probability assigned to each word in test data. Perplexity measures how surprised a language model is when introduced to a new sequence of words. Thus, it can be concluded that a good language model has a lower rate of perplexity.

### 3.3.2. Word Error Rate

Word error rate (WER) is used to measure the performance of a language model in predicting words. This metric is initially used to measure speech recognition models (Klakow & Peters, 2002). WER measures how a language model predicts words in a sequence correctly by comparing predicted words to the original words. WER is obtained by comparing the sequence of predicted words with the original sequence of the words. Equation (16) details how word error rate is measured:

$$WER = \frac{\#\sum_i(\widehat{w}_i \neq w_i)}{N} \qquad (16), \text{ where:}$$

$$\widehat{w}_i = predicted\ word$$

$$w_i = actual\ word\ in\ document$$

$$N = total\ words\ in\ original\ document$$

WER is calculated by comparing predicted words to actual words in the document. If the predicted word is not the same as the original word in the sequence, the numerator will be added by one. The process continues until the end of the document before dividing the sum of errors with the

total words in the original document. Since WER measures how different the prediction result from the original sequence, lower WER indicates better performance/prediction from the model.

Previous research shows that perplexity produced from a word-based language model with finite output linearly correlated to WER (Klakow & Peters, 2002). This means that higher number perplexity is also found on language models with higher WER. This research will focus on analysing the models using perplexity scores because of the design limitation (see Section 2.8.3). Detailed analysis using WER will be done in future research.

# 4. Experiments and Discussion

The section is divided into experiment setup and replication of Vania & Lopez (2017) research, followed by our experiments using Javanese datasets. All our experiments are based on the Vania & Lopez (2017) language model which uses RNN-LSTM architecture (Section 2.8.1). Language model which uses RNN is able to take a sequence of words as an input and use it as a context to predict the next word. While it is superior compared to CNN (Section 2.7), vanilla RNN is prone to wrongly predict a word because the model takes previous information to predict the next word. However, the model could be improved by using the LSTM variant of RNN where LSTM can drop select information from the previous states.

In this chapter, our experiment results will be presented and discussed. This section aims to answer research questions (1) to (4). In order to do that, this section is divided into cross-domain analysis and corrupted model analysis. In the first part of the section, the cross-domain analysis aims to answer research questions (1) to (3). In this part, models will be tested on the three datasets previously described in Section 3.1.1-3.1.3. The aim of this part is leaning towards how the models generalize over different domains, mainly historical and modern datasets. By testing the models using in-domain and out-of-domain tests, we can observe how well the models understands the sequences in the datasets.

The second part of this section is the corrupted model analysis. In the second part, we focus more on research questions (1), (2), and (4). This experiment aims to make the models generalize over the corrupted datasets often found in the OCR documents. Datasets used in this part is AH datasets (Section 3.1.1) alongside its OCR (Section 3.1.4) and manually transcribed counterpart. Manually transcribed dataset means the document that has been manually tagged by humans and used as the most accurate test possible.

The last part of the analysis is by qualitatively analysing the representations using nearest neighbours in vector space. In this section, we aim to answer research questions (1) and (2). The nearest neighbours of words in the vector space will be explored in order to reveal which representation and composition method leads to more intuitive latest representations. The analysis will be done on word-level and subword-level nearest neighbours. The words represented in this section will be analysed semantically and morphologically.

## 4.1. Experimental Setup

In this research, one of our aims is to observe how each model adapted to different domains from the dataset. Our models are trained using three different datasets: Religion and Beliefs (RB; Sastra dataset), Archive and History (AH; Sastra dataset), and Wikipedia. Table 9 details the hyper-parameters that are used to replicate Vania & Lopez (2017) results and our experiments on Javanese datasets.

*Table 9 Model hyper-parameters*

| | |
|---|---|
| **Hidden units** | 200 |
| **Hidden layers** | 2 |
| **Embedding dimension** | 200 |
| **Batch size** | 32 |
| **Time steps** | 20 |
| **Dropout value** | 0.5 |
| **Optimizer** | SGD |
| **Initial learning rate** | 1.0 |
| **Decay rate** | 0.9 |
| **Parameter initialization** | [ -0.1 , 0.1 ] |
| **Word vocab size** | 5000 |
| **Lowercase** | Yes |
| **Number of epochs** | 50 (replication) or 25 |

The experiments will compare language models that operates on the word level (baseline) representations against subword representations. For baseline models, word representations are based on the vector representations of the neighbouring words. Meanwhile, subword representation relies on subwords that compose a word. Using the subword representations (Section 2.7), we compare two composition methods: addition and BiLSTM composition functions.

Figure 12 shows the train and development perplexity for baseline and char-3gram-addition of the Javanese datasets (refer to Section 3.1). As observed, the graphs start to flatten around the 20th epoch. Since there is no significant improvement of training and development perplexity, a maximum epoch number of 25 is applied to all models. This adjustment is considered to be crucial since it reduces the training time while minimally affecting the models.

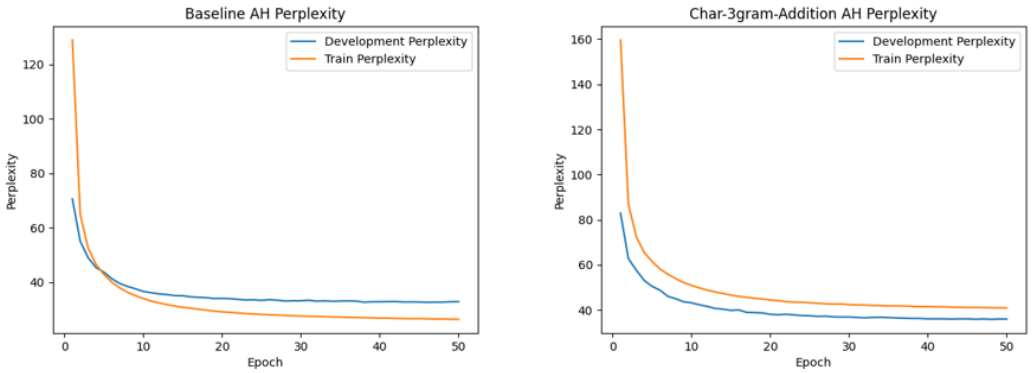



*Figure 12 Train and development perplexity graphs on AH using Baseline and Char-ngram-Addition models*

Three models are trained for each subword segmentation and composition methods combination using the same parameters detailed in Table 9. We report the mean and variance across three runs in order to exclude outliers that might happen if there is only a model trained for each

subword-composition combination. This does not mean that outliers will not happen during three model runs, but its effect could be minimized.

## 4.2. Replication

Before the experiment on the Javanese dataset could be started, results published by Vania & Lopez (2017) needs to be replicated. This step is important to make sure the model works properly, and the design is working as intended. For the replication, a subset of datasets used by Vania & Lopez (2017) and Ling, et al. (2015), namely: English and Turkish datasets are used. Further details about the datasets can be found in Table 10.

*Table 10 Replication datasets summary*

| | **English (Ling, et al., 2015)** | **Turkish (Ling, et al., 2015)** |
|---|---|---|
| **Token** | 1294326 | 657001 |
| **Types** | 84775 | 129979 |
| **Sentences** | 59658 | 47492 |
| **Documents used in train-dev-test set** | 666 | unknown |

Vania & Lopez (2017) used a 0.5 decay rate for their baseline and addition models and a constant 0.2 learning rate for Bi-LSTM models in their experiments. However, the results were unable to be replicated using the specified hyper-parameters in their paper. To make our results comparable to the reproduction, a decay rate of 0.9 to the model is applied while keeping other hyper-parameters to the default number.

Aside from changing the parameters (Section 4.1) to replicate the results, additional preprocessing to BPE segmentation (Section 2.6.2) needs to be added before the model could process the data. Sennrich et al (2016) code separate BPE segmentation with '**@@** ', while Clara & Lopez (2017) input receives BPE segmentation with '**@@**' separator which does not include space. Additional preprocessing removes space from the separator, leaving the segmentations separated by '@@'. For example, the word 'review' becomes '**re@@view**' instead of '**re@@ view**'.

*Table 11 Vania & Lopez (2017) experiment replication*

| | **Baseline (word)** | **Char3gram - Addition** | **Char3gram - BiLSTM** | **BPE - Addition** |
|---|---|---|---|---|
| **English (Vania & Lopez, 2017)** | 46.40 | 45.41 | 42.97 | 47.51 |
| **English (our run)** | 47.40 | 46.743 | 46.919 | 48.477 |
| **Turkish (Vania & Lopez, 2017)** | 66.97 | 50.07 | 54.23 | 59.49 |
| **Turkish (our run)** | 69.204 | 51.63 | - | - |

Based on our replication results in Table 11, we are confident that the code is working as intended. In the next section, we are using the Javanese datasets detailed in Section 3.1 as training data for our models. The models will be analysed based on metrics we have detailed in Section 3.3.

## 4.3. Language Model Performance Within and Across Domains

In this section, we aim to answer the research question (1) to (3), namely: how the models compare when composing a subword representation, what is the best way to compose representations, and how do the models generalize over historical and modern datasets. To answer such questions, two different types of tests are applied to the datasets: in-domain and out-of-domain tests. In-domain tests results will be analysed by testing the model on held-out data from the same domain it was trained on. In Section 4.3.1, models are compared based on their subword representation and composition method in which will compose a word embedding. This way, we can analyse how the models compare also which subword segmentation and composition method performs best in generalizing over datasets. In out-of-domain analysis (Section 4.3.2), models are compared based on how the model generalizes across test sets from different domains (RB, AH, and WIKI). The second analysis is used to answer how the models generalize over different datasets.

The perplexity results of the models are shown in Table 12, Table 13, and Table 14. The results in the tables are grouped based on which dataset they are trained on. Section 4.3.1 In-Domain Analysis will be using Row 3 of Table 12, Table 13, and Table 14, indicated by in-domain on the leftmost column in the tables. Section 4.3.2 Out-of-Domain Analysis will be using Row 4 and 5 from the tables. In the out-of-domain analysis, the models are compared based on the datasets they are tested on. For example, using the RB test set, models that are compared are from Table 12 Row 3, Table 13 Row 4, and Table 14 Row 4.

### 4.3.1. In-Domain Analysis

As observed in Table 13 and Table 14 Row 3, BPE is superior compared to other segmentation for Javanese. This is consistent with previous research (Zhu, Vulić, & Korhonen, 2019b) which shows that BPE holds a significant role in breaking down agglutinative languages. As explained in Section 2.6.2, BPE segments a word based on its most frequent sequence of characters. Since agglutinative languages tend to be regular (Section 2.1), Javanese is easier to model. As a result, models trained on BPE segmentation tend to be less surprised compared to baseline and char-3gram counterparts when exposed to a new sequence.

However, the case is not shared in the RB category (Table 12 Row 3) which shows char3gram to be superior. In Vania & Lopez (2017), which shows char3gram to be superior. In Vania & Lopez (2017), Turkish which is also another agglutinative language shows that char3gram holds a bigger

influence compared to other segmentations. Based on qualitative analysis of the RB dataset, we suspect that the irregularities are caused by mixed language in the dataset. Religion and Beliefs category contains a mix of loanwords which has been described in Section 3.1. This variation of languages increases the possibility of subwords that are not found using BPE segmentation, but better represented using char-ngram segmentation.

From Table 12, Table 13, and Table 14 Row 3, all subword-based models could better represent words when compared to baseline (word) models. The composition method seems to have no systematic difference between BiLSTM and Addition. Historical datasets (RB and AH) show that BiLSTM models have better perplexity compared to addition. However, this fact is not shared with the WIKI dataset which shows that the BPE-Addition model generalizes better for the in-domain test.

The addition method is a simple method that adds all the subword representations in the word to form a word representation. This method is simpler compared to BiLSTM composition that could change word representation drastically. The addition method is a simple method which it adds all the subword representations in the word to form a word representation.

The lower result of the BPE-Addition model might have resulted from the simplicity of the composition function. Since there is a higher number of unknown subwords on the WIKI dataset, embeddings formed using addition profits more compared to using BiLSTM. This is a contrast to AH and RB datasets which are more regular in terms of writing style and fewer loanwords (see Section 3.1.1 and 3.1.2).

*Table 12 Perplexity result trained on Religion and Belief of Sastra dataset using parameters described in Table 9.*

| **Train: RB** | | | | | |
|---|---|---|---|---|---|
| | **Baseline** | **Char3gram** | | **BPE** | |
| | | **BiLSTM** | **Addition** | **BiLSTM** | **Addition** |
| **In-domain** | 47.463 ± 0.631 | **45.594 ± 0.005** | 52.632 ± 3.420 | 60.719 ± 11.698 | 58.485 ± 8.034 |
| **AH** | 55.949 ± 1.500 | 53.508 ± 5.193 | 57.676 ± 4.372 | 60.759 ± 0.898 | 59.502 ± 2.506 |
| **WIKI** | 34.989 ± 4.477 | 31.507 ± 0.436 | 33.209 ± 0.934 | 28.48 ± 0.828 | 28.905 ± 1.586 |

*Table 13 Perplexity result trained on Archive and History of Sastra dataset using parameters described in Table 9.*

| **Train: AH** | | | | | |
|---|---|---|---|---|---|
| | **Baseline** | **Char3gram** | | **BPE** | |
| | | **BiLSTM** | **Addition** | **BiLSTM** | **Addition** |
| **In-domain** | 34.552 ± 0.077 | 35.384 ± 0.930 | 38.304 ± 1.691 | **33.429 ± 0.0009** | 40.246 ± 0.469 |
| **RB** | 57.150 ± 0.940 | 55.198 ± 0.645 | 62.752 ± 2.632 | 56.869 ± 0.722 | 63.123 ± 3.089 |
| **WIKI** | 36.582 ± 2.011 | 30.227 ± 0.472 | 33.361 ± 0.876 | 33.314 ± 2.202 | 30.046 ± 3.595 |

*Table 14 Perplexity result trained on Wikipedia dataset using parameters described in Table 9.*

| **Train: WIKI** | | | | | |
|---|---|---|---|---|---|
| | **Baseline** | **Char3gram** | | **BPE** | |
| | | **BiLSTM** | **Addition** | **BiLSTM** | **Addition** |
| **In-domain** | 24.181 ± 0.019 | 23.825 ± 0.022 | 27.086 ± 0.244 | 24.112 ± 0.211 | **23.582 ± 0.012** |
| **RB** | 56.747 ± 6.383 | 54.172 ± 16.996 | 57.533 ± 4.243 | 44.731 ± 5.119 | 46.097 ± 0.060 |
| **AH** | 46.288 ± 5.591 | 41.911 ± 0.302 | 47.895 ± 56.423 | 35.056 ± 7.382 | 36.927 ± 5.625 |

### 4.3.2. Out-of-Domain Analysis

For the out of domain test, perplexity results are not compared within individual tables. For this test, models are compared based on how well they generalize on different datasets. In order to do that, we need to refer to Rows 4 and 5 of Table 12, Table 13, and Table 14.

Performances are fairly consistent throughout the test sets with in-domain tests producing the lowest perplexity score compared to out-of-domain tests. This result is within our expectations which also shows that our model works properly.

As observed, models trained on WIKI better generalize to AH (Table 14 Row 3) compared to RB models (Table 12 Row 2). Subsequently, WIKI also generalizes to RB (Table 14 Row 2) better compared to AH models (Table 13 Row 2). From perplexity results alone, it seems that WIKI models are less perplexed when exposed to sequences in historical datasets compared to the opposite directions.

Comparing RB (Table 12 Row 3) and AH (Table 13 Row 3) models on the WIKI test dataset, RB models have slightly better perplexity compared to AH. This means that RB models generalize slightly better on WIKI compared to AH. As described in Section 3.1.2, RB contains several different languages. This might be the reason why RB is able to generalize over WIKI which is already known to have a high number of loanwords (Section 3.1.3). With the exposure of loanwords, models trained on RB is able to obtain more subwords combination that is unable to be found in AH.

We suspect that loanwords contained in WIKI play a role in covering information from both historical categories. This is also supported by how RB contains more loanwords compared to AH (see Section 3.1. Section 3.1.2 details composition and languages on the RB test set.) The datasets composition might shed some information about why WIKI models could produce lower perplexity scores on the test set. As observed in Table 8, the number of unique tokens (types) on WIKI datasets is higher compared to other datasets. This high number leads to more variation could be formed by the models during training process. This availability of more subword combination increases the models' understanding and robustness towards unknown words.

Overall, our results show that subwords could better represent words in Javanese compared to word-based representations. In Section 4.3.1, BPE segmentation represents subwords better on AH

and WIKI datasets while Char-3gram represents subwords better on RB datasets. The BiLSTM models are able to perform consistently by having lower perplexity score compared to their addition counterparts across different datasets, especially on Sastra datasets. The only exception to the BiLSTM consistency is BPE-Addition model on in-domain WIKI test.

Section 4.3.2 shows that WIKI models could better generalize over Sastra datasets compared to AH and RB models over one another. We suspect this is because the variety of unique tokens contained the WIKI training set. Higher variety of subword segmentations found in the WIKI dataset contributes to its robustness while generalizing over out-of-domain test sets.

## 4.4. Noisy Input Models Analysis

In this section, we aim to answer research questions (1), (2), and (4): How do models compose word embeddings and could the models generalize over noisy inputs. To achieve that, models are focused on generalizing corrupted test sets. For that reason, only models trained on AH are used in this section. AH dataset is the only dataset used because: (1) it contains documents that are able to be used on the Tesseract OCR, and (2) the AH dataset has the resources which also includes manually transcribed documents.

In this section, models are trained to fit the context of noisy inputs either from OCR mistranslation or mistakes done during manual transcription. The first model is a model trained directly on subword-level. Table 3 shows that words produced from the OCR engines partially matches the manually transcribed documents on character and subword level. Because OCR dataset matches clean texts better on subword-level, models trained in this section are conditioned to receive input from the subword-level dataset. This model will be explained in detail in Section 4.4.1.

The second model is by simulating our datasets to be closer to corrupted texts found in OCR. In this experiment, the synthetic corruption method discussed in Section 2.3 is used. To apply this method, AH datasets are split into lists of five tokens. From these five tokens, one token is randomly chosen and synthetically corrupted using one of three different corruptions: insertion, deletion, and characters transposition. The result of this process is 20% synthetically corrupted AH datasets that will be used as training, development, and test sets.

Our third model is by using the subword regularization method explained in Section 2.6.3 called BPE-Dropout. This method utilizes probability to drop a BPE combination, making it more robust to corruption. Instead of corrupting the dataset like in the second experiment, BPE-Dropout trains the models to be used to partial segmentations which is more representative to the variation found in corrupted sets. Using this method, the models will be able to create representation for incomplete BPE.

In order to answer the questions, we compared four models in this section: vanilla models trained on uncorrupted AH dataset (Section 4.3), subword-level models, SynOCR models, and BPE-

Dropout models. Same as Section 4.3, models are compared using perplexity score by testing on AH, SynOCR, OCR, and manually transcribed test sets. Sections are divided based on how the datasets the models are trained on.

### 4.4.1. Subword-Level Models Analysis

A limitation of the language model design used in Section 4.3 is that it could not form a new word from known subwords. Another design that overcomes this limitation is detailed in Section 2.8.4. The design is able to form new words known words using BPE. However, the open-vocabulary design is not used in this research for the sake of model consistency and comparability.

In this section, models are trained using the word-level composition (baseline) introduced in Section 4.3. Instead of training the model using the subword composition described in Section 2.7, each subword is treated as a word. This method allows the model to overcome its closed-vocabulary design limitation. In addition to that, this method also increases the accuracy of the models by lowering granularity into subwords.

For the OCR test, images provided on the Sastra website are used. A Tesseract OCR engine (refer to Section 2.2) is then used to convert the images into text before they are used as a test set. Perplexity scores from the models are then compared using OCR and manually transcribed test sets. Manually transcribed test sets are taken from the AH test dataset which has been preprocessed using the method described in Section 3.2.

In this section, both OCR and manually transcribed test sets are preprocessed by splitting the words found in the document into their subwords. The additional preprocessing step is only done to the test sets so that the models would be able to recognize the tokens on which units the models are trained. In this case, test documents split using char-ngram segmentation would only be tested using models trained on char-n-gram with baseline composition. The same is also applied to BPE which is also included as one of the segmentations in this experiment.

#### *OCR Perplexity Analysis*

Table 15 shows how models perform on different test documents. From the perplexity result, the Baseline-Word model produces lower perplexity on both OCR and manually transcribed test sets.

#### OCR Test Set

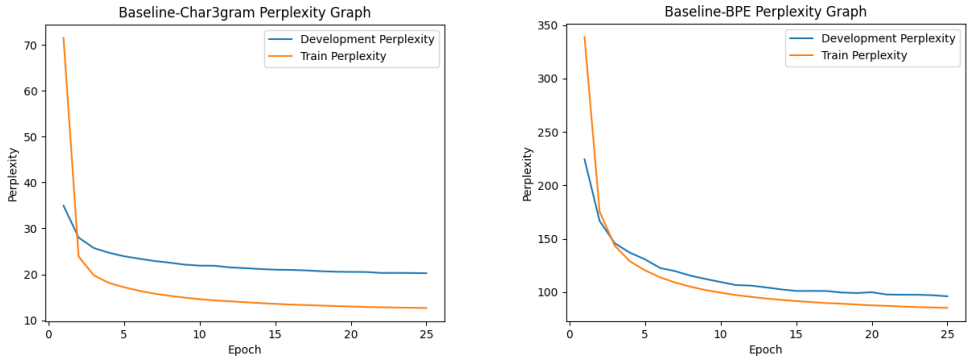
Based on the perplexity result (Table 15 Column 3), the Baseline-Word model produces the lowest perplexity result compared to the other Baseline-Char3gram and Baseline-BPE. As explained in Section 2.8.3, the language model replaces words that appear once in the training with unknown tokens. Since the OCR test set tends to have a higher rate of corrupted and segmented tokens, the Baseline-Word model replaces the corrupted representation with unknown token representation

during the prediction. This means that instead of generalizing over known words, the Baseline-Word model generalizes over unknown tokens on the OCR test set. This later is reflected by the model having lower perplexity because the OCR test set sequence is translated into unknown token followed by unknown token.

The reason models with BPE segmentation shows poor perplexity result is because it failed to understand the sequence. Since BPE highly relies on the longest combination of characters in a token, it is possible for BPE to predict the wrong subword if the data were to be corrupted. If we were to corrupt the word ***ditumbasaken*** from the previous example into ***d1tumbasaken***, instead of splitting the word into [***d1, tumbas, aken***], BPE will split the word into [***d1tumbas, aken***]. In this case, instead of using representations for [***<unk>, tumbas, aken***], the model uses [***<unk>, aken***] which further confuses the model since it has never been encountered during the training.

The Char3gram model also shows similarly poor perplexity results on the OCR test set. The model seems to be unable to generalize over the OCR test set. However, as observed in Figure 13, the models manage to converge during the training and development. This might be caused by the low similar documents' usage in the training set. Since similar documents are rarely encountered during training, the model is unable to generalize over such documents once found.

*Figure 13 Training and Development perplexity graph of Baseline-BPE and Baseline-Char3gram models*




## Manually Transcribed Test

This test is done on manually transcribed OCR document discussed in previous section. The result of this test could be seen in Table 15 Column 4. In this experiment, the Baseline-Word model produces lowest perplexity score compared to other models. This means that model is able to generalize over manually transcribed test set better than subword-level models. Because the test set is more similar to training set compared to the OCR test set, the Baseline-Word model has a good chance to have the words in the manually transcribed set recorded in the vocabulary. Thus, the model is less perplexed when exposed to the sequence in the test set.

### AH Test Set

As observed in Table 15 Column 2, the model trained on char-3gram subword-level (Table 15 Row 2) has better perplexity compared to other models. There are several reasons why the subword-level model is better for this case. The first reason is that the AH test set closely resembles the training set compared to OCR and manually transcribed test sets which are shorter. Therefore, the model would be able to generalize better over the AH test set.

The second reason is that because char-3gram tend to be more regular compared to BPE and word. Table 5 in Section 2.6.4 shows that word ***ditumbasaken*** is split into [***dit, itu, tum, umb, mba, bas, asa, sak, ake, ken***] which follows [**123, 234, 345, …**] pattern. Because of the pattern created by char-n-gram segmentation, it is easier for the model to predict the subword that comes next.

*Table 15 Perplexity result of OCR and manually transcribed counterpart trained using baseline composition*

| Model | AH | OCR | Manually transcribed |
|---|---|---|---|
| **Baseline – Char3gram** | **22.024** | 105.903 | 105.609 |
| **Baseline – BPE** | 95.556 | 370.569 | 205.334 |
| **Baseline – Word** | 34.236 | **22.283** | **65.416** |

BPE segmentation in this experiment has much higher perplexity compared to other segmentations. This is a contrast compared to the result produced in Table 13 which shows BPE-BiLSTM having the best perplexity.

Since Javanese morphologically is an agglutinative language, it tends to rely on affixes to modify words. If the word ***ditumbasaken*** from Table 5 is split into its main word and affixes using BPE segmentation, it will form [***di, tumbas, aken***]. Compared to affixes that are often encountered during training, base words such as [***tumbas***] will likely be replaced with unknown token because it is rarely encountered. Ultimately, the model failed to understand which affix should be used based on a specific context in the sequence because base words are replaced with unknown representations unless they are often encountered.

Overall, all models trained using subword-level datasets are unable to generalize over the OCR and manually transcribed test sets as indicated by the comparatively higher perplexity values. This experiment shows that the Baseline-Word model which is also evaluated in Section 4.3 to better generalize over noisy inputs. However, the Baseline-Char3gram model shows its advantage on the AH test set. Compared to the Baseline-Word model, the Baseline-Char3gram model has a lower granularity and more regularized sequence as previously discussed in the AH Test Set section. This resulted in the Baseline-Word model being more perplexed because of less regularity on the word-level dataset and its bigger vocabulary size.

### 4.4.2. Synthetic OCR Analysis

We generate synthetic OCR (SynOCR) to simulate corrupted data often found in the OCR dataset. This method usually is used to supplement the lack of an OCR dataset (see Section 2.3). Ideally, SynOCR models would be developed and tested on an OCR dataset. In this experiment, SynOCR is used as train, development, and test sets because of a lack of suitable OCR documents in Sastra datasets (refer to Section 3.1.4).

For this experiment, SynOCR corruption (Section 2.3) is applied to AH datasets. How SynOCR corruption is generated has been discussed in earlier part of Section 4.4. To compare the models, three models have been trained on the SynOCR dataset: baseline (word) composition, char-3gram with addition composition, and char-3gram with BiLSTM composition. In this section, models are compared based on how they perform on different test sets namely: SynOCR, OCR, and manually transcribed.

#### *SynOCR Perplexity Analysis*

Table 18 Columns 3-5 shows model perplexity results on different test sets: SynOCR, OCR, and its manually transcribed counterpart. Based on the perplexity result, models trained on SynOCR (Table 18 Rows 5-7) have comparable score compared to models trained on manually transcribed counterparts (Table 18 Rows 2-4). In this section, models compared are from Table 18 Rows 2-7 Columns 3-5.

*Table 16 Perplexity result of SynOCR models in comparison to their vanilla counterparts*

| **Train: AH** | **SynOCR** | **OCR** | **Manually transcribed** |
|---|---|---|---|
| **Baseline** | 52.142 ± 0.744 | 22.620 ± 0.033 | 65.669 ± 18.218 |
| **Char3gram-Addition** | 37.907 ± 0.398 | **18.620 ± 0.670** | 67.694 ± 9.584 |
| **Char3gram-BiLSTM** | 54.617 ± 1.558 | 20.979 ± 0.091 | **58.401 ± 11.834** |
| **Baseline (trained on SynOCR)** | 41.290 ± 0.033 | 32.338 ± 22.619 | 105.780 ± 10.410 |
| **Char3gram-Addition (trained on SynOCR)** | 39.668 ± 0.008 | 18.920 ± 5.844 | 76.763 ± 2.830 |
| **Char3gram-BiLSTM (trained on SynOCR)** | 36.922 ± 0.104 | 21.804 ± 0.496 | 72.076 ± 12.653 |

### OCR Test Set

Char3gram-addition model trained with non-corrupted dataset produces the best perplexity among models trained on SynOCR (Table 16 Rows 5-7) and vanilla models (Table 16 Rows 2-4) when tested on OCR Test set (Table 16 Column 3). However, the perplexity of the Char3gram-addition model trained with a corrupted dataset also shows a perplexity result that is not too far behind. Even though all models show lower perplexity scores than manually transcribed test set, it is questionable whether they capture the OCR dataset properly. Models might replace subword representations with unknown token representations since the respective subword was not included in the vocabulary.

### Manually Transcribed Test Set

From Table 16 Column 4, models trained on non-corrupted datasets (Table 16 Rows 2-4) shows the lowest perplexity when exposed to the manually transcribed test set. This result is not surprising considering that the test is done "in-domain" compared to models trained on the SynOCR dataset (Table 16 Rows 5-7). Char3gram-BiLSTM model shows the best perplexity compared to other models. This result mirrors the in-domain test shown in Table 13 which shows Char3gram-BiLSTM as the third-best model after BPE-BiLSTM and Baseline model.

Among models trained on the SynOCR dataset, the Char3gram-BiLSTM shows the best perplexity on the manually transcribed test set. This result perfectly mirrors its non-corrupted model counterpart discussed in the previous paragraph. It seems that the BiLSTM composition function benefits more from subword units. Char3gram-BiLSTM models seem to be able to generalize better compared to baseline (word) and addition counterparts. This is also demonstrated in other historical datasets (Table 12 and Table 13 Row 1) where models with BiLSTM composition generalize better compared to other compositions.

### SynOCR Test Set

Based on Table 16 Column 2, the baseline model trained on SynOCR produces lower perplexity compared to its non-corrupted counterpart. This is not surprising considering that it is an "in-domain" test for models trained on the SynOCR dataset (Table 16 Row 5-7). Similar to the situation in the manually transcribed test, subword units combined with BiLSTM composition shows much better performance compared to other models.

Perplexity results obtained from the Char3gram-addition model trained on uncorrupted train set are best among all models trained on this dataset. The reason why the addition model has lower perplexity lies in its simplicity. When forming a word representation from subwords, addition models would sum subword representations created during training. This process typically does not have as much effect compared to BiLSTM which might change the word representation drastically. Using the addition model, the effect of unknown subword representations could be reduced.

It could be said that this experiment improved generalization on the OCR test set. Models trained on SynOCR datasets performs better on synthetic OCR test sets while not losing much when tested on manually transcribed test sets. However, SynOCR datasets might not be a good fit for models to train for to generalize over the OCR test set. As observed in Table 3, OCR documents have a higher rate of corruption which might not be able to be completely substituted by SynOCR. The difference between OCR and SynOCR opens up possibilities for future work by either incorporating OCR documents as development set or using additional embeddings (März et al., 2021).

Our language models (Section 2.8.3) replace words with a single occurrence encountered during training with an unknown token. This could be one of the reasons why the models fail to understand the importance of corruption during training. Further explained in Section 2.3 about SynOCR sets generation, words are randomly corrupted by inserting, deleting, or transposing random characters. Our method of random corruption means that there is a very low chance of the same corruption occurring more than once. Because a single occurrence of the token is replaced with unknown tokens, SynOCR models could not record the corrupted tokens in the datasets into the vocabulary. This means that SynOCR might not be an ideal setup for the model to generalize over noisy inputs.

The OCR contains more corruption compared to SynOCR datasets (see Table 3 for dataset comparison). Since the baseline models could not find the representation of corrupted tokens in the vocabulary, the model replaces it with unknown word representation. This resulted in the model later recognizing the test set as a sequence of unknown tokens. The OCR perplexity results tend to have lower perplexity score because most tokens are replaced with unknown tokens. In the case of subword models, the model constructs word embedding using unknown subword representation because of unknown trigrams that are found in the OCR test set.

Numbers produced by models trained on SynOCR and tested on OCR test set indicates that the model does not learn much from the SynOCR dataset. This indicates that models trained on SynOCR might not be a good fit to learn corruption from. März et al (2021) point out the usage of SynOCR without any embeddings sometimes decreases performance because the model misrecognizes tokens in the document. Since there are no Javanese subword embeddings used as a reference, SynOCR models learn the incorrect representation during training.

Overall, in this experiment char-3gram segmentation seems to benefit most in predicting OCR and manually transcribed test sets. The result is not surprising considering we have seen many occasions of subword models superior when compared to their word-based counterparts (Kim et al., 2016; Bojanowski et al., 2017; Vania & Lopez, 2017). This result also further emphasizes that char-3gram is more robust to corrupted datasets. In addition to that, addition composition seems to be able to generalize over different datasets; improving perplexity results on corrupted test sets, while not losing much when tested on the manually transcribed test set.

This experiment partially answers our research questions (2) and (4). The char-3gram segmentation with addition composition method trained using vanilla dataset performs best in this experiment. The model performs well on noisy and vanilla test sets. The perplexity score of this model shows that it could generalize well on SynOCR even without introducing noises in the training phase.

### 4.4.3. BPE-Dropout Analysis

In this section, we use BPE-Dropout in order to improve models' robustness when exposed to corrupted test sets (Provilkov et al., 2020). For this experiment, the BPE-Dropout train set is tuned and tested on AH datasets used in previous BPE testings (Section 4.3). Similar to SynOCR tests, BPE-Dropout models are also tested on the OCR dataset and its manually transcribed counterpart.

In Section 2.6.3, BPE-Dropout is described as a variety of BPE which has a probability to drop subword combinations. This process is done to simulate corruption that often happens in OCR documents. Similar to what has been done in previous research (Provilkov et al., 2020), a dropout probability of 0.1 is applied to the training documents. This means that words in the training document have one in ten chance not to have the longest sequence as BPE usually split documents, but instead they use incomplete morphemes. Since incomplete morphemes are often found in noisy datasets, the models would be able to generalize noisy inputs such as OCR documents better. Figure 4 details an example of the difference between BPE and BPE-Dropout splitting where BPE-Dropout splits demonstrate shorter segmentations compared to BPE.

Similar to BPE, BPE-Dropout models split words based on its segmentations created during the BPE application. In this research, words are split using design[3] created by Sennrich et al. (2016) which implements dropout method proposed by Provilkov et al. (2020). Similar to how other word embeddings are created, subword representations from BPE and BPE-Dropout uses a composition method (Section 2.7) to form a word representation.

BPE-Dropout models are trained over BPE-Dropout train sets and tested on a normal BPE set. Since BPE-Dropout splits words into incomplete BPE, it is not reasonable to develop the embeddings using the BPE-Dropout development set as well. BPE-Dropout models are supposed simulate a noisy segmentation found in corrupted documents. Tuning the subword representation using BPE-Dropout set makes the model overfits into BPE-Dropout context. To prevent that, the models are tuned using BPE without dropout which would be the case for test sets. This way, BPE-Dropout can capture the actual meaning of affixes (corrupted or non-corrupted).

Compared to SynOCR, there would be more BPE-Dropout subword combinations recorded in the dictionary. Referring to Section 2.8.3, the language model stochastically replace words with a single occurrence to an unknown token which makes most of the synthetically corrupted words from

[3] https://github.com/rsennrich/subword-nmt

SynOCR be not recorded in the dictionary. However, in BPE-Dropout there is a higher chance for dropout combinations to make it to the vocabulary, especially for frequent subwords. Because the words frequently appear in the document, it is not hard for BPE to recognize the pattern and create a segmentation out of it. The higher the number of words in the document, the more combinations and multiple entries of the said combinations could be created. Thus, lowering the chance of the words to be replaced by unknown represenations. This means there would more subword variations in the BPE-Dropout vocabulary which is preferable for noisy contexts such as OCR.

### *BPE-Dropout Perplexity Analysis*

Table 17 details perplexity results obtained from BPE models. In this table, the result from BPE without the dropout method is also included as a comparison. The test sets used in the table consists of original AH test sets, previously used in Section 4.3 alongside the OCR and manually transcribed test sets.

*Table 17 Perplexity results of BPE-Dropout models in comparison to their BPE counterparts*

| Train: AH | AH | SynOCR | OCR |
|---|---|---|---|
| **BPE-Addition** | 40.246 ± 0.469 | 15.788 ± 0.244 | 88.971 ± 36.919 |
| **BPE-BiLSTM** | 33.429 ± 0.0009 | 20.967 ± 0.076 | 59.538 ± 13.560 |
| **BPEDropout-Addition** | 33.204 ± 0.112 | **12.540 ± 0.515** | **18.236 ± 6.327** |
| **BPEDropout-BiLSTM** | **26.919 ± 0.019** | 56.724 ± 11.482 | 48.538 ± 35.439 |

#### OCR Test Set

In the OCR test set (Table 17 Column 4), the BPEDropout-Addition model (Table 17 Row 3) has the lowest perplexity compared to other models. This means that the dropout method implemented in BPE effectively introduces noise in a way that is useful for modelling noise as found in the Tesseract OCR set. Compared to the BPEDropout-BiLSTM model (Table 17 Row 5), the addition composition method better generalizes over corrupted datasets. This is also an occurrence that happens in Section 4.4.2, which shows the addition model to be superior in the OCR test set.

#### Manually Transcribed Test Set

On the manually transcribed documents (Table 17 Column 4), both BPE-Dropout models (Table 17 Rows 4-5) are superior compared to their non-dropout counterparts. The BPEDropout-Addition model (Table 17 Row 4) also performs better on manually transcribed test set compared to its BiLSTM counterparts. It might be easier for the document to generalize over manually transcribed

and OCR document because of: (1) its simplicity in forming word representations and (2) the model does not completely capture the context of the test set because it only makes up a small percentage in the training set (refer to Section 3.1.4).

### AH Test Set

In addition to the previous two test sets, the BPE test set (Table 17 Column 2) which is used previously in Section 4.3 is also used in this section. Based on the perplexity result in Table 18, BPE-Dropout models (Table 17 Rows 4-5) generalize better over different test sets in this analysis. Compared to other test sets which shows that addition composition has better generalization, this test set shows that the BPEDropout-BiLSTM model better generalizes over the AH test set. However, the fact that the addition composition result is not too far behind shows BPE-Dropout units is able to cover more representation compared to BPE.

In this test set, compared to addition, the BiLSTM composition (Table 17 Rows 3 and 5) shows better generalization compared to the addition counterparts from both units. As previously discussed in the Manually Transcribed Test Set of Section 4.4.3, the document group that becomes the manually transcribed and OCR test sets made up of a small percentage in the training set. This means that the models lack a source to learn from the training set. This in turn makes the models produce more unstable perplexity scores when exposed to the test sets because the document is rarely encountered during training.

Perplexity is calculated based on negative average log probability $H$ (refer to Section 3.3.1). Because of how the perplexity score is calculated, there is a possibility of the low exposure to sequences found was accounted as an outlier by the models. This is the reason for the models having rather unstable perplexity scores when exposed to the OCR and manually transcribed test set, but not on the AH test set which is more similar to the training and development sets.

*Table 18 Perplexity results on manual transcription and OCR test set*

| Train: AH | OCR | Manually transcribed |
|---|---|---|
| **Baseline** | 22.620 ± 0.033 | 65.669 ± 18.218 |
| **Char3gram - Addition** | 18.620 ± 0.670 | 67.694 ± 9.584 |
| **Char3gram - BiLSTM** | 20.979 ± 0.091 | 58.401 ± 11.834 |
| **Baseline (trained on SynOCR)** | 32.338 ± 22.619 | 105.780 ± 10.410 |
| **Char3gram - Addition (trained on SynOCR)** | 18.920 ± 5.844 | 76.763 ± 2.830 |
| **Char3gram - BiLSTM (trained on SynOCR)** | 21.804 ± 0.496 | 72.076 ± 12.653 |
| **BPE - Addition** | 15.788 ± 0.244 | 88.971 ± 36.919 |
| **BPE - BiLSTM** | 20.967 ± 0.076 | 59.538 ± 13.560 |
| **BPEDropout - Addition** | **12.540 ± 0.515** | **18.236 ± 6.327** |
| **BPEDropout - BiLSTM** | 56.724 ± 11.482 | 48.538 ± 35.439 |
| **Baseline – Char3gram** | 22.024 | 105.903 |
| **Baseline – BPE** | 95.556 | 370.569 |

From Section 4.4, subword regularization method performs better compared to other methods: subword-level models and training dataset corruption models. However, the subword-level models failed to generalize over the shorter documents such as OCR and manually transcribed documents. This is because the document group used in the test set makes up a small percentage of the training set. The AH test set which has more similar sequences to the training and development sets produces lower more stable perplexity score (Table 18 Column 3).

Perplexity scores of models trained using the SynOCR datasets (Table 18 Rows 5-7) are higher compared to other models. This could be caused by the models being tuned using the SynOCR datasets instead of the OCR development set. This makes the models being unable to generalize on the OCR test set. Several proposed adjustments might be used to improve the perplexity results. The first is by implementing additional embeddings as proposed by März et al. (2021). The second reason is by tuning the models using the OCR development set. Since there is a low resource of documents that could be OCR'ed using Tesseract, documents that are written using traditional Javanese script (Carakan) could be OCR'ed using Trawaca OCR[4]. However, the OCR engine for Carakan is still in its

[4] https://trawaca.id/ocrjawa/

early stage of development and can only digitize Javanese scripts without translating them to Latin alphabets. The digitized scripts later need to be transcribed either manually or using an engine into the Latin alphabets before it could be used as a development set.

The subword-level training sets do not give any improvement to the models. The Baseline-Word model performs better on the manually transcribed and OCR test sets, indicated by lower perplexity scores in Table 15 Columns 3-4. As previously discussed, both test sets are less similar as the documents only make up a small percentage of the training set. This might be the cause of why subword-level models are unable to generalize over the test sets. The statement is supported by the Baseline-Char3gram model result which produces lower perplexity on the AH test set (Table 15 Column 2) which is closely resembled the training set.

## 4.5. Nearest Neighbours Analysis

In this section, we aim to qualitatively answer question (1) and (2) namely how words representations compare with each other, and what is the best segmentation and composition method combination to compose Javanese. In order to answer those questions, word representations are analysed qualitatively using nearest neighbours. The section is split into two different ways in analysing nearest neighbours: word-based and subword-based analysis.

The term “nearest neighbours” in NLP means closest objects to a certain point in vector space. Points in this vector space are created from the representation vectors calculated during training. The baseline model creates the representation of words. However, subword models compose subword representations instead of saving word embeddings as is. Word embeddings from subword models are then calculated based on one of the composition methods explained in Section 2.7 every time it encounters a new test set.

There are two types of similarities that could be used to describe nearest neighbours: semantic and morphology. Morphologically similar words means that words have same base word with different affixes. Semantically similar words are words that have different bases but have similar meanings or are synonyms.

### 4.5.1. Word Nearest Neighbours

In this section, the nearest neighbours will be analysed based on their segmentation and composition methods. As explained before, the baseline model saves word embeddings as is while subword models save subword representation instead. For models in Table 19, Table 20, and Table 21, embeddings are extracted from the embeddings layer in the models. In the design by Vania & Lopez (2017), settings in baseline and BiLSTM allow word embeddings to be extracted directly from the model. However, such is not the case for addition composition. We therefore minimally modified the code as follows.

To obtain the word representation from addition composition, the embeddings layer containing subword representations first need to be extracted. Words in the vocabulary dictionary are later split into their subwords. Word representations are then calculated by summing all subword representations extracted from the word. This way, word embeddings for addition models could be obtained without drastically changing the original code.

Words in Table 19, Table 20, and Table 21 are chosen randomly from commonly used words in Javanese. After the word representations are extracted from the models, nearest neighbours are analysed using Embedding Projector[5] using cosine distance. Nearest neighbours are translated through a combination of a Javanese speaker (Nathaniel) and Educalingo dictionary[6].

The reference words in Table 19, Table 20, and Table 21 are placed on the first row of each table. Five closest words from the referenced words in the vector space are then presented in the block underneath the reference words. For example, in Table 19 five nearest neighbours obtained from the baseline model of word ***kaping*** (number -) are [***ping, pukul, suwidak, jam, angaping***]. In Table 20, the same reference words are used. However, the model is further split into how the word embeddings are composed separated by table blocks, either using the addition or BiLSTM composition. Similar to Table 20, Table 21 also shows the nearest neighbours of the referenced words using the addition and BiLSTM composition methods. The difference between Table 20 and Table 21 lies in which subword segmentation are used: char-3gram in Table 20 and BPE in Table 21.

As shown in Table 19 and Table 20 Column 3, baseline and char-3gram models could better represent words semantically than the BPE segmentations (Table 21). For example, word ***dhawuh*** (order) has a close relationship with ***utusan*** (delegate) on baseline and char-3gram-BiLSTM models. This is different from BPE representations which tend to represent words based on their morphological composition (Table 21).

By comparing the nearest neighbours of word ***dhatêng*** (to) in Table 19, Table 20, and Table 21 Column 4, how different segmentations react to the word could be observed. In Table 19, the word ***dhatêng*** (to) has a closer relationship to another word ***mênyang*** (to) compared to the form of ***dhatêng*** (to). Meanwhile, in Table 21, the same word has a closer relationship to its variations rather than the word ***mênyang*** (to) which also has the same meaning.

BPE segmentation tends to represent words morphologically because it splits words based on common patterns (see Section 2.6.2). Since agglutinative languages use a set of affixes to modify a word, models trained on BPE segmentation tend to put more importance on a word compared to affixes. It resulted in BPE segmentation to be able to represent morphological variations of a word compared semantically similar words.

[5] http://projector.tensorflow.org/
[6] https://educalingo.com/en/dic-jv

*Table 19 Word level nearest neighbours of words in Javanese using Baseline trained on AH*

| Model | jv word | en translation | jv word | en translation | jv word | en translation | jv word | en translation | jv word | en translation |
|---|---|---|---|---|---|---|---|---|---|---|
| | **kaping** | **number -** | **dhawuh** | **order (n)** | **dhatêng** | **to ; onto (pp)** | **dhahar** | **to eat (v)** | **tiyang** | **person (n)** |
| **Baseline** | ping | number - ; times | nawala | newsletter | marang | to head | tindak | to go | têtiyang | people |
| | pukul | - o'clock ; to hit | dhawah | da'wah; missionary endeavor | mring | to head | kaparêng | got permission | rare | child |
| | suwidak | Sixty | andhawuhakên | ordered to | dhumatêng | to ; onto | karsa | to work | lare | child |
| | jam | - o'clock ; time | têdhak | Stepping down | maring | to head | rata | even ; flat | waraha | boar (avatar of Vishnu) |
| | angaping | | utusan | delegate | kalihan | with | kadarman | good (action) | priyantun | nobles |

*Table 20 Word level nearest neighbours of words in Javanese using Char-3gram segmentation trained on AH*

| Model | | jv word | en translation | jv word | en translation | jv word | en translation | jv word | en translation | jv word | en translation |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **kaping** | **number -** | **dhawuh** | **order (n)** | **dhatêng** | **to ; onto (pp)** | **dhahar** | **to eat (v)** | **tiyang** | **person (n)** |
| **Char-3gram** | **Addition** | gumaruduk | come with crowd | bedol | (not found) | taraton | (not found) | rèhantya | (not found) | dandani | dressing up ; fixing |
| | | magêpok | acquainted | têngên | right | ruba | insincere | keliatan | (able to be) seen | gantungan | hanger |
| | | piniyagah | doing according to own terms | beginsel | (not found) | pohung | cassava | kasilip | | mêngani | Open (personality) |
| | | ngiringan | escort | tatunirèng | (not found) | lèitu | (not found) | kliwone | first day in Javanese week ; village head | jantur | to hang ; hanging |
| | | pinayungan | using umbrella | overal | (not found) | verplichte (Dutch) | blamed | anêbah | beat | ridêr | (not found) |
| | **BiLSTM** | ping | number - ; times | têdhak | stepping down | marang | to head | kagyat | surprised | pamulangan | return |
| | | pukul | - o'clock ; to hit | utusan | delegate | dhumatêng | to ; onto | duta | delegate | têtiyang | people |
| | | fatsal | | bilah | blade | mênyang | to head | brana | there | rare | child |
| | | artikel | article | budhale | departure | maring | to head | brani | brave | tiyangipun | the person |
| | | f | F | dhawah | da'wah; missionary endeavor | mring | to head | brangkat | to go; to head | rat | (not found) |

*Table 21 Word level nearest neighbours of words in Javanese using BPE segmentation trained on AH*

| Model | | jv word | en translation | jv word | en translation | jv word | en translation | jv word | en translation | jv word | en translation |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **kaping** | **number -** | **dhawuh** | **order (n)** | **dhatêng** | **to ; onto (pp)** | **dhahar** | **to eat (v)** | **tiyang** | **person (n)** |
| **BPE** | **Addition** | kacêkaping | enough | dhêdhawuh | order | dhatênging | coming of | dhahara | to eat (imperative) | tiyanga | person |
| | | kapingx | number - ; times | dhawuha | to order (imperative) | dhatênga | to come (imperative) | adhahar | to eat | tiyang9 | Person |
| | | kapingv | number - ; times | dhawuhi | Ordered (by) | dhatêngi | to come (passive) | dhar | to open ; to birth | tiyang5 | Person |
| | | kapingul | (not found) | dhawuhe | Order (of) | dhatêngpl | to come | dhahare | food of - | tiyangka | person |
| | | kapingit | (not found) | adhawuh | To order | dhatêng21 | to come | dhaharan | food | tiyangingkang | The person who - |
| | **BiLSTM** | ping | number - ; times | cawuh | repeatedly | dhumatêng | to ; onto | dhahas | dry | têtiyang | people |
| | | pining | (not found) | rawuh | to go (home) | mênyang | to ; onto | dhawak | alone | tiya | person (corrupted) |
| | | ajaping | (not found) | 3dhawuh | Order | dhumatê | to ; onto (corrupted) | dhawah | da'wah ; missionary endeavor | tiyo | (not found) |
| | | angaping | (not found) | adhawuh | To order | mênyan | incense | dhêrak | big gathering | sayang | love ; empathy |
| | | adêgkaping | arranged number - | 2dhawuh | order | marang | to head | dhiyar | | tiyang9 | person |

### 4.5.2. Subword Nearest Neighbours

In this section, the nearest neighbours of BPE subwords will be analysed by comparing the same words previously used in Table 19, Table 20, and Table 21. Since these words are frequently used, there is a high chance for the words to appear among the BPE subwords. The reason why char-3gram is not included in the subword comparison is that char-3gram subwords consist of three consecutive characters. The segmentation would be less meaningful because it only presents trigram segmentations unlike BPE which resembles morphemes.

BPE-Dropout models have been excluded from word-level nearest neighbours analysis because the models tend to make different embeddings for the same word. However, we include this model in our subword nearest neighbours analyses.

Table 22 shows the nearest neighbours of frequently used words in Javanese. In the table, the nearest neighbours shown are subword representations in the embeddings layer along with their English translation whenever applicable. Note that some subwords are incomplete due to BPE segmentation, indicated by a hyphen (-) before or after the word.

Different from word-based embeddings where BPE tend to have morphologically similar words, BPE subword representations represent words semantically. Take an example from the word ***tiyang*** (person) in the BPE-BiLSTM model. The model represents the word closer to "people", "child", "warrior", and "whoever". This means that the model understands the relations between said words better compared to its addition counterpart which is unable to represent the complete and meaningful sentence.

The trait of representing semantically similar words is also shared by the BPEDropout-BiLSTM model. The model represents most of the words that are semantically similar to the selected word. In addition to that, the BPE-Dropout model is able to represent a corrupted word as observed under the word ***tiyang*** in Table 22 Row 5 of the BPEDropout-BiLSTM model section (morpheme ***tiy-*** which is a part of the word). Observing nearest neighbours in Table 22 also reveals that BPE-Dropout models include incomplete morphemes, especially under the word ***dhahar*** (to eat). One of the examples is ***-êdhe*** which might be a part of word ***gêdhe*** (big).

This shows that BPE-Dropout is more robust because it could represent incomplete and less frequent subwords which are often found in corrupted documents. By using incomplete subwords, the models could combine the representations to better generalize over more unobserved subwords in the test sets, both corrupted and manually transcribed sets.

All words in Table 22 could be represented either semantically or morphologically quite well by the models. However, there is an exception to this, in which the word ***dhahar*** seems to be represented by either incomplete word or semantically. There is a reason why this could happen in the nearest neighbour representation. One of them being the word is rarely used in the document. A quick token check using a python script reveals in the AH datasets, subword ***dhahar*** appears 416 times while subword ***dhawuh*** (order) appears 2080 and ***tiyang*** 2892 times. This is understandable because

the word ***dhahar*** is often used in everyday conversations but rarely appears in official documents (refer to Section 3.1.1).

The low number of word ***dhahar*** occurrences means that the model lacks evidence to learn the word/subword representation from. This forces the model to learn the representation from neighbouring words or subwords. This is also demonstrated in Table 21 where the word ***dhahar*** is closest to other words that are morphologically similar.

Overall, from the nearest neighbours analysis, we could gain qualitative insights which can help answer research questions (1) and (2). Based on the analysis in Section 4.5.1, baseline and char-3gram models could represent words semantically. This is because the models take information from the surrounding word and create a representation around it. BPE models, however, tend to represent words morphologically. This is because the models rely heavily on the morphemes created from BPE segmentations.

In addition to that, the BiLSTM composition method also demonstrates a better representation of words using different segmentations. Compared to its addition counterpart, BiLSTM models combine subword representations into more meaningful and related words. This could be observed in Table 20, where the Char3gram-BiLSTM (Table 20 Row 3) model produces more words with related meanings compared to the Char3gram-Addition model (Table 20 Row 2).

Different from its word-level nearest neighbours, in Section 4.5.2, BPE segmentation could also represent subwords semantically. This further emphasizes that models with BPE segmentation benefit more from the morphemes that compose a word. Instead of using morphemes, subword-level nearest neighbours force the BPE to represent words/subwords based on the surrounding morphemes.

Section 4.5.2 also shows how words are represented using BPE-Dropout subword regularization. The models manage to create more partial morphemes compared to BPE. This partial segmentation later increases the variety of morphemes that the models might encounter on noisy test sets. BPE-Dropout models also demonstrate high robustness by generalizing over noisy inputs as discussed in Section 4.4.3.

*Table 22 Subword level nearest neighbours in Javanese using BPE segmentation trained on AH*

| Model | | jv word | en translation | jv word | en translation | jv word | en translation | jv word | en translation | jv word | en translation |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **kaping** | **number -** | **dhawuh** | **order (n)** | **dhatêng** | **to ; onto (pp)** | **dhahar** | **to eat (v)** | **tiyang** | **person (n)** |
| **BPE** | **Addition** | obah - | to move | - kale - | | - itu | (not found) | - kuli | (not found) | rider | (not found) |
| | | ngir - | | têngên | right (direction) | - joem - | (not found) | sat - | (not found) | kagarwa | (not found) |
| | | sajatining | actually (personality) | - adhuh | (exclamation) | pinjam | to borrow | solah | (not found) | - poj - | (not found) |
| | | - pantês | fit | - bilih | (not found) | purang | (not found) | - li - | (not found) | bikak | to open |
| | | blz | | - lu - | (not found) | warn- | (not found) | - êdhe | (not found) | terse - | |
| | **BiLSTM** | ping | number - | têdhak | to step down | marang | to head | duta | representative | têtiyang | people |
| | | fatsal | | utusan | delegate | dhumateng | to ; onto | kagyat | surprised | rare | child |
| | | pukul | to hit ; - hours | tindak | to go | kalihan | with ; second | mila | beginning | wadya | warrior |
| | | jam | time; - hour | wangsul | answer | mring | to head | putri | female ; princess | singa | Whoever ; lion |
| | | artikel | article | lilah | permission | maring | to head | gêntos | turn | kadang | rarely |
| **BPE Dropout** | **Addition** | niet | | nyangga | (not found) | wal | (not found) | - among | (not found) | kathen | (not found) |
| | | angge | for | - êkok | (not found) | prentah - | order (n) | - ingah | (not found) | - ilir | (not found) |
| | | ruk - | (not found) | - inggu | (not found) | kad - | | - pira | how many | katata | exemplary action (corrupted) |
| | | - êgara | (not found) | mangidul | to the south | pangendh - | order (corrupted) | - utawi | or | keris | traditional weapon |
| | | nyum - | (not found) | material | material | - + - | | kasoran | lose | lêluhur | ancestor |
| | **BiLSTM** | fatsal | (not found) | bantu | to help | marang | to head | - iri | (not found) | tiy - | person (corrupted) |
| | | ping | number - | pitulung | help | dhumatêng | to head | - ti | (not found) | têtiyang | people |
| | | artikel | article | tindak | to go | mring | to head | amung | but | rare | child |
| | | nomêr | number - | usul | suggestion | mênyang | to go | wadya | warrior | tiyangipun | the person who |
| | | pukul | to hit ; - hours | rawuh | to go home | awasta | (not found) | - ud | (not found) | wadya | Warrior |

# 5. Conclusion

There is a lack of NLP research on Javanese in general and especially in historical Javanese. In Section 1, it is established that there is a need to increase the resources for future studies. In recent studies, an increasing trend of subword embeddings could be detected compared to word embeddings. Subword embeddings are preferred over word embeddings because they could supplement the lack of resources in low-resource languages. Thus, this research aims to understand how subword representations can improve language models for historical Javanese. Can a model train on modern Javanese and understand historical Javanese? Which subword representations and composition method could best represent historical Javanese? In addition to that, we also aim to know whether the models are able to understand corrupted contexts that are often found in historical Javanese due to OCR errors or transcription mistakes.

In Section 2, previous findings are discussed. In the section, Javanese as a language is discussed morphologically and grammatically. Different styles of Javanese used in historical and modern contexts are also compared in the section. In the same section, the OCR dataset and synthetic corruption is also discussed alongside how simple domain adaptation works. In the later part of Section 2, different types of word representations are introduced. Word representations are used as input to the language model which becomes our focus in this research. Various language models are described based on the design and their limitations. At the end of the section, we discussed which model fits our needs best alongside our consideration of choosing one over another.

Section 3 details the datasets used in this research. This section introduces our first contribution: a curated version of Sastra datasets covering historical documents from different domains. The data have not previously been used in NLP research. The datasets are also compared based on years they were written, language style, and similarity. We also detailed further dataset preprocessing to make the datasets similar to previous research. Evaluation metrics used in this research are also detailed in this section.

In Section 4, experiment results and discussions are presented. We started the chapter by detailing hyperparameters used in this research. We managed to replicate the previously presented results to ensure that the design is working properly. In later sections, models are analysed based on how they perform on different test sets.

The first experiment that was done is a cross-domain experiment (Section 4.3). In this section, models are compared using in-domain and out-of-domain analysis. The cross-domain experiment aims to answer the research questions (1), (2), and (3). In order to do that, the experiments focus on how models trained using different word representations compares to one another. The in-domain analysis similarly shows that subword models have lower perplexity compared to their baseline counterparts. The results are consistent with previous research which also shows that subword models could better represent words. In our experiment, the BPE segmentation performs better on historical

and modern datasets, represented by the AH and WIKI test sets. The BiLSTM composition method also shows best performance when it is used to generalize over historical datasets, while the addition composition method performs best on modern datasets. Through this experiment, it is also shown that the models trained on WIKI datasets could generalize quite well on Sastra (historical) dataset.

The AH and WIKI models similarly show that BPE segmentation holds higher importance compared to other segmentations. This is in line with previous research by Zhu, Vulić, & Korhonen (2019b) which also shows BPE segmentation to be superior in agglutinative languages. However, the RB model produces lower perplexity for char-3gram segmentation with BiLSTM composition. The RB model leans toward char-3gram because different loanwords that it contains (Section 3.1.2). Nevertheless, the result is also in line with another research (Vania & Lopez, 2017) which shows char-3gram to be superior in Turkish.

The second experiment focuses on how the models generalize to corrupted datasets. This experiment focuses on the research questions (1), (2), and (4). In this experiment, analyses are split based on which dataset the models are trained on. We experimented on three different types of model training: making the training set into subword-level set then testing the document on subword-level using baseline composition, training the models using synthetically corrupted datasets, and BPE-Dropout subword regularization models.

In the first part of the experiment, we tried to overcome design limitations by splitting the words into subwords. The AH datasets are split into their subwords before models are trained on the datasets. The AH test set shows that the Baseline-Char3gram model can best generalize by showing a lower perplexity score. On the OCR test set, the Baseline-Word model shows lower perplexity. However shown in Section 4.4.1, the model generalizes over unknown tokens instead of proper words by assigning unknown tokens as the true label to the OCR sequence. This explains why the model has an unusually lower perplexity score when compared to the manually transcribed test.

In the second part of the experiment, models are trained on the SynOCR dataset. This dataset simulates corruption as explained in Section 2.3. The result shows that SynOCR models do not have the robustness it is intended to have. März et al (2021) detail that the use of SynOCR without additional embeddings sometimes have a negative result. In our experiment, models trained on SynOCR datasets are unable to generalize over the OCR and manually transcribed test sets properly. Subword models trained on the AH datasets generalizes better on the OCR and manually transcribed instead.

In the third part of the second experiment, models are trained using BPE-Dropout segmentation. The experiment shows that BPE-Dropout models are superior compared to their BPE counterparts. Models trained on BPE-Dropout segmentation could better generalize over the AH, OCR, and manually transcribed test sets. It is also shown that the addition composition method performs better on shorter documents while BiLSTM on longer ones.

The final part of our experiment is by qualitatively analysing the nearest neighbours of the words in vector space. This section is split into two: word and subword nearest neighbours. In word nearest neighbours, models with BPE segmentation tend to put morphologically similar words closer compared to semantically similar. This is because BPE segmentation relies on the most frequently observed sequence in the words (Section 2.6.2). This means that BPE is better suited to identify affixes that are often found in agglutinative languages.

The second part of the nearest neighbours analysis is by comparing subwords formed using BPE and BPE-Dropout. Char-3gram is not included in this analysis because the segmentation is unable to make a meaningful sequence. Different from word nearest neighbours, subword nearest neighbours analysis shows that both BPE and BPE-Dropout are able to find semantically similar words.

# 6. Future Works

There are limitations in this research, one of them being the models unable to create words out of the vocabulary composed during the training phase. We tried to overcome this limitation by using the subword-level models. However, the models are proven to be unable to generalize over subword-level test sets. There are also several methods that we have yet to experiment on. One of the proposed methods is by increasing the value of output vocabulary size. Using this hyperparameter tweaking, we could hypothetically increase the accuracy of the predictive model. However, this might in turn worsen the perplexity score and increase the training time for the models because there are more output words for the model to choose from.

In Section 4.4, the models are trained to generalize over noisy inputs. Overall, subword regularization method, BPE-Dropout, models perform best over different test sets (corrupted and non-corrupted). However, the models trained using synthetically corrupted datasets (SynOCR) could not perform well on both test sets. This problem is caused by the lack of documents that could be read using the Tesseract OCR engine. The limitation makes the model unable to be tuned over the OCR development set. Instead, it is tuned using the SynOCR development set which proves to be an unideal setup. We can tackle this by adding OCR documents for SynOCR models development.

Another alternative that could be explored is by using the Trawaca OCR engine. The engine digitizes Javanese scripts into their digital encodings which allow computers to read the script. Since the engine is still in development, it is still unable to transcribe the script into Latin alphabets. However, the digitized Javanese script could be transcribed either manually or automatically into Latin alphabets before using it as corrupted development or test sets. This way, the lack of corrupted datasets could be partially solved using additional steps.

Another method that could be used is by using the subword-based language model design (Kim G. , 2019) described in Section 2.8.4. However, the performance of the model for a longer sequence has yet to be explored. The design initially is used for query completion which is often used for search engines. Incorporating the design with our language model seems to be an interesting direction to pursue in the future.

One evaluation metric that has not been explored is WER evaluation (Section 3.3.2). WER evaluation is excluded from this research due to the language model design limitation. Subword models are able to generalize over different subword combinations. However, it is unable to find the suitable word from the finite dictionary, thus, making WER results worsen by predicting unknown tokens instead.

# References


Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., . . . Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Al-Rfou', R., Perozzi, B., & Skiena, S. (2013). Polyglot: Distributed Word Representations for Multilingual NLP. *Proceedings of the Seventeenth Conference on Computational Natural Language Learning* (pp. 183–192). Sofia, Bulgaria: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/W13-3520

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics, Volume 5*, 135–146. doi:10.1162/tacl_a_00051

Cieri, C., Maxwell, M., Strassel, S., & Tracey, J. (2016, May). Selection Criteria for Low Resource Language Programs. *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, 4543–4549. Retrieved from https://www.aclweb.org/anthology/L16-1720

Eisenstein, J. (2018). *Natural Language Processing.* MIT Press. Retrieved from https://github.com/jacobeisenstein/gt-nlp-class/blob/master/notes/eisenstein-nlp-notes.pdf

Errington, J. J. (1992). On the Ideology of Indonesian Language Development: The State of a Language of State. *Pragmatics, 2*(3), 417-426.

Gage, P. (1994). A new algorithm for data compression. *C Users Journal, 12*(2), 23-38.

Gerz, D., Vulić, I., Ponti, E., Naradowsky, J., Reichart, R., & Korhonen, A. (2018). Language Modeling for Morphologically Rich Languages: Character-Aware Modeling for Word-Level Prediction. *Transactions of the Association for Computational Linguistics, Volume 6*, 451–465. doi:10.1162/tacl_a_00032

Grave, E., Bojanowski, P., Gupta, P., Joulin, A., & Mikolov, T. (2018). Learning Word Vectors for 157 Languages. *arXiv preprint arXiv:1802.06893*.

Graves, A., Fernández, S., & Schmidhuber, J. (2005). Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition. *International conference on artificial neural networks* (pp. 799-804). Berlin, Heidelberg: Springer.

Grunwald, P. (2004). *A tutorial introduction to the minimum description length principle.* doi:arXiv:math/0406077

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation, 9*(8), 1735–1780. doi:https://doi.org/10.1162/neco.1997.9.8.1735

Jones, K. S. (1972). A Statistical Representation of Term Specificity and Its Application in Retrieval. *Journal of Documentation, 28*, 11-21. doi:10.1108/eb026526

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). Bag of Tricks for Efficient Text Classification. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers* (pp. 427–431). Valencia, Spain: Association for Computational Linguistics.

Kim, G. (2019). Subword Language Model for Query Auto-Completion. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International*

*Joint Conference on Natural Language Processing* (pp. 5022–5032). Hong Kong: Association for Computational Linguistics.

Kim, Y., Jernite, Y., Sontag, D., & Rush, A. M. (2016). Character-Aware Neural Language Models. *arXiv:1508.06615v4*.

Klakow, D., & Peters, J. (2002). Testing the correlation of word error rate and perplexity. *Speech Communication, 38*, 19-28. doi:10.1016/S0167-6393(01)00041-3

Krisnawati, L. D., & Mahastama, A. W. (2018). A Javanese Syllabifier based on Its Orthographic System. *2018 International Conference on Asian Language Processing (IALP)* (pp. 244-249). Bandung, Indonesia: IEEE.

Krisnawati, L. D., & Mahastama, A. W. (2019). Building Classifier Models for on-off Javanese Character Recognition. *iiWAS2019: Proceedings of the 21st International Conference on Information Integration and Web-based Applications & Services*, (pp. 25-34). doi:https://doi.org/10.1145/3366030.3366050

Kudo, T. (2018). Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates. *arXiv preprint arXiv:1804.10959*.

Labeau, M., & Allauzen, A. (2017, September). Character and Subword-Based Word Representation for Neural Language Modeling Prediction. *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, 1–13. doi:10.18653/v1/W17-4101

Ling, W., Dyer, C., Black, A. W., Trancoso, I., Fermandez, R., Amir, S., . . . Luís, T. (2015). Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 1520–1530). Lisbon, Portugal: Association for Computational Linguistics. doi:10.18653/v1/D15-1176

Luhn, H. P. (1957). A Statistical Approach to Mechanized Encoding and Searching of Literary Information. *IBM Journal of Research and Development, 1*(4), 309-317. doi:10.1147/rd.14.0309

März, L., Schweter, S., Poerner, N., Roth, B., & Schütze, H. (2021). Data Centric Domain Adaptation for Historical Text with OCR Errors. *International Conference on Document Analysis and Recognition*, (pp. 748-761).

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Karafiát, M., Burget, L., Černocký, J., & Khudanpur, S. (2010). Recurrent Neural Network Based Language Model. *INTERSPEECH*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.

Ogloblin, A. K. (2006). Javanese. In K. A. Adelaar, & N. Himmelmann (Ed.), *The Austronesian Languages of Asia and Madagascar* (pp. 590–624). London and New York: Routledge.

Provilkov, I., Emelianenko, D., & Voita, E. (2020). BPE-Dropout: Simple and Effective Subword Regularization. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 1882–1892. doi:10.18653/v1/2020.acl-main.170

Ramponi, A., & Plank, B. (2020). Neural Unsupervised Domain Adaptation in NLP—A Survey. *CoRR*. Retrieved from https://arxiv.org/abs/2006.00632

Sennrich, R., Haddow, B., & Birch, A. (2016). Neural Machine Translation of Rare Words with Subword Units. *arXiv preprint arXiv:1508.07909*.

Smit, P., Virpioja, S., Grönroos, S.-A., & Kurimo, M. (2014). Morfessor 2.0: Toolkit for statistical morphological segmentation. *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics* (pp. 21–24). Gothenburg, Sweden: Association for Computational Linguistics. doi:10.3115/v1/E14-2006

Suharno, I. (1982). *A Descriptive Study of Javanese.* Canberra: ANU Asia-Pacific Linguistics / Pacific Linguistics Press. doi:doi:10.15144/PL-D45

tesseract-ocr. (2021, September). *Languages supported in different versions of Tesseract.* Retrieved October 2021, from Tesseract documentation on GitHub: https://tesseract-ocr.github.io/tessdoc/Data-Files-in-different-versions.html

Vania, C., & Lopez, A. (2017). From Characters to Words to in Between: Do We Capture Morphology? *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 2016–2027). Vancouver, Canada: Association for Computational Linguistics. doi:10.18653/v1/P17-1184

Wedhawati, N., & Arifin, S. (2006). *Tata bahasa Jawa mutakhir [A contemporary grammar of Javanese] (in Indonesian).* Yogyakarta: Kanisius.

Yayasan Sastra Lestari. (2021, April 26). *Sastra Lestari: Profil*. Retrieved from Sastra Jawa: https://www.sastra.org/tentang/sastra-lestari-profil

Zhu, Y., Heinzerling, B., Vulić, I., Strube, M., Reichart, R., & Korhonen, A. (2019a). On the Importance of Subword Information for Morphological Tasks in Truly Low-Resource Languages. *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)* (pp. 216–226). Hong Kong, China: Association for Computational Linguistics. doi:10.18653/v1/K19-1021

Zhu, Y., Vulić, I., & Korhonen, A. (2019b, June). A Systematic Study of Leveraging Subword Information. *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)*, 912–932. doi:10.18653/v1/N19-1097