Data-driven Supervised Morpheme Extraction

Ramya P Narayanaswamy, Dariusz Kuc, Chris Kim

Dept. of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois

{rpn2, dkuc2, ykim157}@illinois.edu

ABSTRACT

With the ever-increasing availability of large unstructured data and the need to extract useful information, text segmentation is one of the predominant research topics in data mining. Some of the key applications of text segmentation are natural language processing, topic modeling, machine translation etc., Various forms of text segmentation exist, and popular segmentation techniques are focused on character sequences, words, phrases, and sentences. However, text Segmentation to extract morphemes, which are smallest meaningful unit of a language has proven to improve effectiveness of downstream data mining tasks. Most of the previous research in morpheme extraction is unsupervised with the best accuracy of ~20%. We model morpheme extraction as a classification task with data-driven statistical features, resulting in an accuracy of ~65%.

KEYWORDS

Segmentation, subwords, morphemes, supervision, classification, probabilistic random forest

1 INTRODUCTION

Today's world is becoming more connected with ever-increasing number of electronic devices. The connected world generates roughly 2.5 quintillion bytes a day. Almost 90% of data in today's world is created in the last few years [1][2]. A significant portion of such data is unstructured text. In recent years, data mining community has been focused on research efforts related to mining unstructured data. Information extraction from unstructured text is a critical task for many applications including but not limited to entity typing and recognition, sentiment analysis, natural language processing, machine translation, topic analysis, contextual advertising etc.,

Text segmentation is a fundamental step in structuring text data. It is a process of dividing text input into a sequence of characters, words, phrases and sentences. The segmented text is used as features for data mining tasks like classification, clustering, outlier detection etc., For instance, entity recognition could be modeled as a classification task with machine learning models generated by neural networks or random forests. Previous efforts in entity recognition used word-based features, character-based features or combination of both [3]. However, one could consider adding additional features that are finer granularity than words and coarser granularity than sequence of characters to further improve the classification accuracy. The insight is if "claustrophobia" is given as entity mention, then "aerophobia" could be recognized as entity mention as they share same subword "phobia". Such subwords are smallest meaningful units of a language and are called morphemes. Morphemes could be regarded as different parts of the word: roots, prefixes, and suffixes. Every word typically consists of one or more morphemes.

Traditionally, NLP tasks utilize word and/or character-based feature embeddings [4] [5] [6] [7]. Word embeddings have been shown to also perform significantly better in a variety of NLP tasks when morpheme information is considered [8] [9]. A significant insight is that many morphemes provide informative hints as to the part-of-speech of a word. Given that POS taggers are an inexpensive set of tools, it is possible to derive the likely part of speech of a word from individual subwords. This can be used to enrich particular subwords with POS information and percolate down the pipeline to enrich word embeddings. Furthermore, morphologically rich languages such as Indo-European languages often form words using smaller morpheme substructures. One area that may particularly benefit from utilizing morphemes is machine translation whereby these subword structures may aid in proper alignment between source and target languages in parallel corpora. Furthermore, morphemes provide a computationally inexpensive way to promote parameter sharing among words, and address word sparsity and out of vocabulary issues in machine translation [10] [11].

Thus, morphemes are powerful units of a language that could improve the effectiveness of downstream data mining tasks. The fundamental step to enable such improvement is extraction of morphemes from a given text corpus. Most of the previous techniques for morpheme extraction are unsupervised [12] [13]. We derive motivation for using data-driven supervised technique based on improved performance of supervised phrase mining approaches [14] [15] over unsupervised approaches [16] [17] for data mining applications. The main contributions of our research project are as follows

I. Morpheme extraction is modelled as a classification task where every split point of a word is given a label 0 (no-split) or 1 (split). For instance, for the word *"aerophobia"*, in-order labels are [0 0 0 1 0 0 0 0 0]. The '1' at index 4 indicates that the split point for the word is after character 'o', generating *"aero"* and *"phobia"* as morphemes.

II. A set of statistical and character n-gram features for each split point are defined. A probabilistic random forest classifier provides the probability of each index being a split point. i.e. for the word *"aerophobia"*, we invoke Random Forest classifier 10 times for each split point to generate a "0" or "1" label, based on thresholding of output class probability.

III. Our framework is domain independent and language agnostic w.r.t feature generation and classification as opposed to some of the previous work [18] [19] in morpheme extraction.

IV. We demonstrate the proof-of-concept and effectiveness of our framework by comparing the accuracy of morpheme extraction against state-of-the-art unsupervised methods using DBLP title dataset.

The rest of the paper is organized as follows. Section 2 is a background study of different types of morphemes and some of the notable work in morpheme extraction. Section 3 describes our morpheme extraction framework and features used for classification. Section 4 details our experimental setup, comparison with unsupervised techniques, and analysis of the results. Section 5 presents a few suggestions for future improvement, and Section 6 provides conclusion.

2 BACKGROUND

2.1 Types of Morphemes

Our framework could extract morphemes that exist in various structural forms. This section briefly describes popular forms of morphemes. Morphemes could be primarily categorized into two types: unbound and bound morphemes. Unbound or free-form morphemes function independently as words or could appear with other morphemes as lexemes. In English, words like *"chair"*, *"man"* and *"ship"* are termed as unbound morphemes as they could occur as independent words. Words like *"townhall"* (town + hall), *"dollhouse"* (doll + house) are instances of unbound morphemes that co-occur with other morphemes.

Bound morphemes appear only as part of the words, always in conjunction with root of a word or with other bound morphemes. For instance, in English, affixes like *"un- "*, *"im- "*, *"re- "*, *"-ous"*, *"-tion"* are bound morphemes that always occur in conjunction with a root. Bound morphemes are further categorized into inflectional and derivational morphemes. Inflectional morphemes influence the base words to signal a change in quantity or tense with similar meaning as base word. Some examples of inflectional morphemes in English are *"-s"*, *"-ed"*, *"-ing"* suffixes that are added to a root. (Example: race to races, raced, racing). Comparative morphemes *"-er"*, *"-est"* (Example: big to bigger, biggest) are part of inflectional morphemes. On the other hand, derivational morphemes change the function or meaning of the root word. Some examples of derivational morphemes in English are prefixes like *"un- "*, *"in- "*, creating words like *"unhappiness"*, *"uncool"*, *"insufficient"*, *"inability"*.

2.2 Previous work in morpheme extraction

A brief history of notable morpheme extraction techniques is summarized below. Some of the techniques provide an insight to define the features for our classification framework. Our background study also reveals that some of the morpheme extraction techniques are domain and language dependent.

2.2.1 Unsupervised morpheme extraction using statistical methods

Creutz et al. [12] describes a couple of unsupervised techniques for extracting morphemes using statistical methods. These techniques are language agnostic and authors demonstrated the principles with both English and Finnish. The first method uses Minimum Description Length (MDL) and recursive segmentation for morpheme discovery. MDL is used as the cost function and the objective is to find optimal segmentation that provides the lowest cost function. The data model of a given corpus consists of vocabulary of morphs termed as code-book and the objective is to find morphemes that produce a concise code-book. The total model cost is summation of Cost(Source text) and Cost(Codebook) as in Equation 1. Cost of source text is negative likelihood of the morph. Cost of codebook is number of bits required to represent each character in every morph of the code book.

$$\begin{aligned} C &= Cost(Source\ Text) + Cost(Codebook) \\ &= \sum_{tokens} -\log\ p(m) + \sum_{types} k * length(m) \end{aligned} \tag{1}$$

An online recursive segmentation algorithm is used for morpheme segmentation. For every word in the corpus, various segmentation options are evaluated and the one that lowers the cost function is eventually chosen as the split point. Recursive splitting searches for the split point in a top-down fashion starting from a complete word. A hierarchical tree like data structure is used. The segmentation algorithm selectively removes previously encountered sub-words/words from the tree and recomputes morphemes for every instance of words, thus avoiding local optima. Additionally, the algorithm improves sub-optimal segmentation of less optimal words by periodically re-segmenting the previously segmented data.

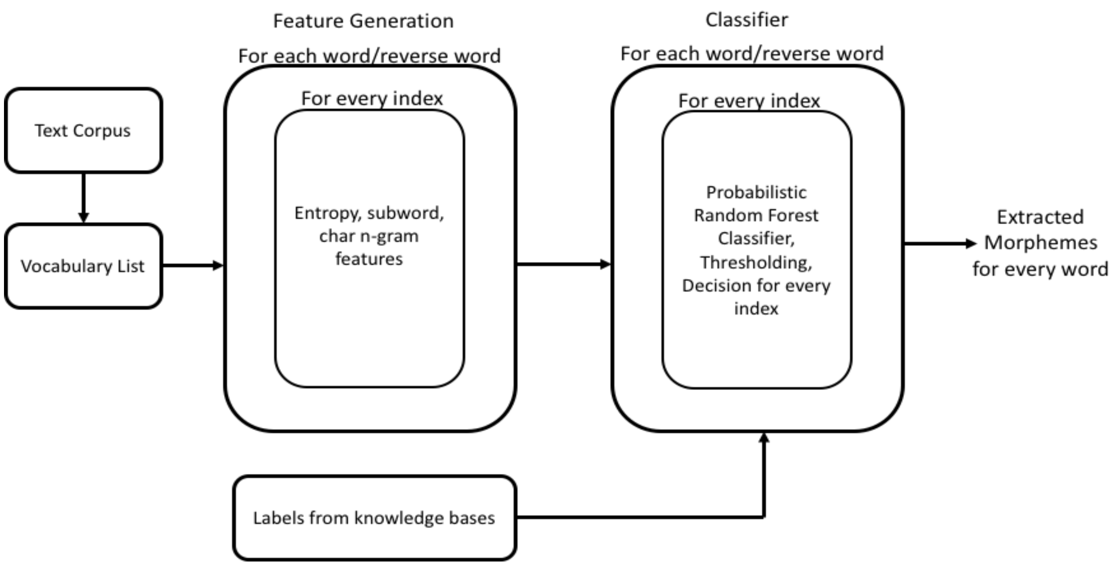

Feature Generation
For each word/reverse word
For every index
Entropy, subword, char n-gram features
Classifier
For each word/reverse word
For every index
Probabilistic Random Forest Classifier, Thresholding, Decision for every index
Text Corpus
Vocabulary List
Extracted Morphemes for every word
Labels from knowledge bases

Figure 1: Data-driven supervised morpheme extraction framework

The second method deploys sequential segmentation and Maximum Likelihood (ML) Cost. The algorithm uses ML as cost, which is the first term *Cost(Source text)* in Equation 1. An offline batch learning Expectation-Maximization (EM) like Viterbi algorithm is deployed for optimizing the cost function and segmentation is linear. The initial segments or various morphs of every word are obtained in a random fashion using Poisson distribution. At every iteration, morph probabilities are estimated, Viterbi algorithm is used for finding the segmentation that yields lowest cost for each word. At the end of every iteration, rare morphs and segments that yield two or more one letter morphs are rejected and random initialization occurs for words whose morphs have been rejected.

Bernhard [13] proposes an unsupervised technique for morphological segmentation based on segment predictability and word segments alignment. The first step is segment predictability to identify a set of prefixes and suffixes for a given corpus. This is accomplished by calculating transition probabilities of substrings at every position of the word. Given a word w of length n, S_{ij} represents the substring starting at index i and ending at index j. The mean of maximum transition probabilities for all substrings starting and ending at each position k of the word is calculated using Equation 2. The frequency of a substring is the number of times it occurs in the given corpus.

$$f(k) = \frac{\sum_{i=0}^{k-1} \sum_{j=k+1}^{n} \max\left[p\left(S_{ik}|S_{kj}\right), p\left(S_{kj}|S_{ik}\right)\right]}{k * (n-k)} \qquad (2)$$

The profile of variation of transition probability is used to determine local minima. The position at which local minima occurs indicates a potential segment boundary. Stem is identified as the longest segment, substrings that precede and follow stem are termed as prefixes and suffixes. The following step involves word segmentation by comparing words that have same stem to find limits between shared and different segments. This step also identifies additional prefixes, suffixes and linking elements associated with the text corpus. Word alignment-based segmentation generates multiple segmenting options (one segment per stem) for a given word. A list of best segments or morphemes for each word is generated based on segment frequency in the entire text corpus.

2.2.2 Morpheme Extraction for Neural Machine Translation

Neural Machine Translation (NMT) is one the popular ways to translate a text or speech from one language to another using Recurrent Neural Networks(RNN). The recurrent neural network consists of encoder-decoder network to aid in translation. The bidirectional encoder network processes an input sequence to obtain a fixed-size annotation vector. The decoder network predicts a target sequence based on previously predicted word, context vector and hidden state of RNN. Typically, stochastic gradient descent is used for training and a beam search is employed for translation. Traditional NMT's operate on word-level models and addresses the translation of out-of-vocabulary and rare words by backing off to a dictionary.

Rico Senrich et al [10] propose an NMT that uses a morpheme-based model in which rare and unknown words are encoded as a sequence of morpheme units, eliminating the need for large back-off dictionaries. A Byte Pair Encoding (BPE) is proposed for achieving subword segmentation. BPE helps in achieving the goal of rare words translation through fixed-size vocabulary and variable length character sequences. The primary motivation of using subwords translation is that it helps achieving direct translation of named entities, cognates and loanwords, and morphologically complex words and generalizes this knowledge to translate unseen words. BPE learns the vocabulary of the given text that produces a good compression. BPE iteratively replaces the most frequent pair of characters with a character sequence. Each merge operation produces a new n-gram character sequence (termed as symbol). Thus, BPE constructs a final symbol vocabulary from a given source text, where number of merge operations is a hyperparameter. The character sequences generated by BPE are morphemes, which serve as input to neural networks for translation. Two BPE methods were devised for comparative purposes. The learning of encodings of source and target vocabulary are done independently in the first scheme and they are done jointly in second scheme (joint BPE).

Taku Kudo proposes a subword segmentation for NMT based on unigram language model [11]. The unigram model assumes that subwords occur independently and output probabilities for a given subword sequence is product of individual subword occurrence probabilities. The most probable segmentation for any input sequence is typically obtained by solving for maximum likelihood using EM algorithm, if the vocabulary size is known in advance. However, in most real datasets, vocabulary size is unknown and is learnt from the dataset. Taku Kudo uses an iterative algorithm to tackle the joint optimization of vocabulary size and their occurrence probability. The iterative algorithm is initialized to a seed vocabulary. Some of popular techniques used to enumerate initial seed are union of all characters and frequent substrings in the dataset. EM algorithm is repeated until desired vocabulary size is obtained. Thus, unigram language model produces a subword segmentation which is a probabilistic mixture of subwords, characters and words.

2.2.3 Morphemes in medical document retrieval

The internet has paved the way to collect many GB's of medical information and make them readily accessible to millions of health care professional and patients. However, searching and retrieving medical documents has been a challenging task. This is primarily due to complex morphological structure associated with medical sublanguage that renders simple string matching-based algorithms to be insufficient. Udo Hahn et al [18] propose a scheme where both the input query and the set of documents are subject to subword segmentation prior matching process. The matching process effectively retrieves all the relevant documents that not only contains all the documents that have the query words in direct form, but also various documents that contain the morphological variants of the input query.

Traditional linguistic approaches consider morphemes as smallest meaningful entities and approach the segmentation based on this assumption. In contrast, in medical sublanguages, subwords form the smallest meaningful unit as the meaning of a combination of linguistically significant morphemes is almost equal to another non-decomposable medical synonym. The algorithm described in this paper utilizes a medically meaningful subword segmentation by creating dictionaries of subwords, prefixes, infixes, derivational and inflectional suffixes, which are several orders of magnitude less than full lexicalized dictionaries. The segmentation engine utilizes the above dictionaries as resources and concatenation regularities to segment the query and document collection. Ambiguous segmentations are ranked based on longest match from left, minimal number of stems per words, minimal number of consecutive affixes, segmentation weight factor of subwords and affixes.

2.2.4 Lexical morpheme segmentation system for Chinese

Chinese language poses many challenges to the text segmentation tasks. With thousands of characters and without explicit cues such as inflexion and capitalization performing lexical analysis of a Chinese text is extremely challenging task. In this paper by Fu, et. al. [19], researchers propose a framework that utilizes morphemes to combine text segmentation task and part-of-speech tagging into a single task. Proposed algorithm works as follows, given a sentence in Chinese, morpheme segmentation splits the text into a sequence of morphemes using provided dictionary. Since there could be multiple ways on how to segment a sentence, algorithm combines information from bigram language models with morpheme formation models to resolve ambiguities. This information is then used by lexical chunking component that labels each morpheme with a tag indicating its position within a word, i.e. whether the morpheme is at the beginning of the word, inside the word, at the end of the word or if the morpheme is an individual word by itself. Post processing is applied to transform the output to a sequence of part-of-speech tagged words.

From the background study, it is evident that the existing morpheme segmentation techniques are unsupervised, and some of them are domain and language dependent [18] [19].

3 Supervised Morpheme Extraction

Our framework for supervised morpheme extraction is shown in Figure 1. The fundamental blocks of our framework are vocabulary extractor, feature generator and classifier. The vocabulary extractor creates a list of unique words from a text corpus. Resulting list of words is normalized by removing all non-alphanumeric characters, and then converted to lowercase. Vocabulary extractor can also optionally filter its output by removing common stop words from the results.

During the feature generation phase, every word is processed individually and multiple features for each split point are generated. The features can be grouped into substring transition entropy, substring count and character n-grams categories. The substring transition entropy and substring count features are derived based on principles described by unsupervised morpheme discovery by Bernhard [13]. Bernhard suggests that substring transition probability of a word are good indicators of morpheme boundary. An index in a word is likely to be a morpheme boundary if a local minima of substring transition probability occurs at the index. In our framework, we identify the transition points of a word from the substring transition entropy and substring count. A notable difference between Bernhard's method and our features is that only substrings that start at index *0* and end at index *k*, where *k* is *1* to length of the word are considered, whereas all substrings of a word are considered in Bernhard's method.

For classification, a probabilistic random forest classifier is used. Labelled data for segmentation is generated from a knowledge base. Labelled data provides expected root words/morphemes for a subset of words in the vocabulary list. Index of the morpheme substring within a word is used as an identifier to encode the label information. As a result, each character split point of a word is given a label 1 (split) or 0 (no- split). For instance, label encoding for *"aeroplane"* is [0,0,0,1,0,0,0,0,0], indicating the split point is after character 'o'. The resultant morphemes are "aero" and "plane". This labelled data set is used for training.

The random forest classifier is trained on every split point for all the words in the training data. During testing phase, classifier outputs probability for each index of a target word. Final label, i.e. split point decision is chosen based on the configurable threshold. A downside of configurable threshold is that many split points could be generated for a target word, thereby causing over-splitting. This is overcome by limiting the maximum number of split points to 2 by choosing two highest probabilities that satisfy the threshold. Two split points are required as it helps to identify compound words that have both prefix and suffix. For instance, the word *"unhappiness"* would have 3 subwords *"un", "happi", "ness"*, thus necessitating 2 split points.

A detail description of features used for classification is presented below

3.1 Substring transition entropy features

The substring entropy features are directly influenced by El-Kishky's research on entropy-based subword mining for word embeddings [20]. The substring transition entropy of a string X is defined in Equation 3, where $f(*)$ represents the substring count in the vocabulary list.

$$H(X) = -\sum_{i} \frac{f(Y_i)}{f(X)} \log \frac{f(Y_i)}{f(X)} \tag{3}$$

Y_i is a substring of length $M + 1$. X is a substring of length M, where first M characters match with Y_i. For each potential split point of the word associated with index k, three transition entropies over the substring starting at index *0* and ending at index *k-1*, *k* and *k+1* are calculated. By including entropy of a substring at a position before and after the current index, the classifier could potentially identify the transition in substring entropy and generalize towards determining the split index of a given word.

As the substring always starts at index *0*, the conditional entropies defined above are intended to identify split points for words that have common prefixes. For instance, words like *"aerobics", "aeroplane", "aerospace"* have common prefix *"aero"*. To account for words that have common suffixes, like *"-ed"* as in *"waited", "raced", "gathered"*, for any given index, three additional entropy features from reversed words is calculated. As a result, six entropy-based features are generated for every potential split point.

aeroplane	enalporea
a eroplane	enalpore a
ae roplane	enalpor ea
aer oplane	enalpo rea
aero plane	enalp orea
aerop lane	enal porea
aeropl ane	ena lporea
aeropla ne	en alporea
aeroplan e	e nalporea
aeroplane	enalporea

Figure 2: Split points of "aeroplane"

3.2 Substring count features

Substring count features represent the raw count of substrings in the entire vocabulary for every word starting at index *0* and ending at index *k*, where *k* ranges from *1* to length of the word. In total, six substring count features for every potential split point of a word including three features based on prefixes and three features based on the suffixes (i.e. prefixes of reversed word) are generated.

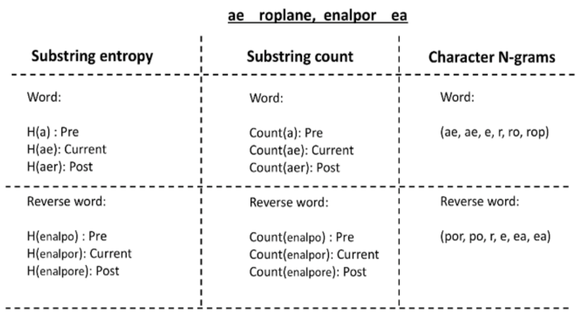
3.3 Character n-grams features

For every potential split point, unigram, bigram and trigram characters to the left and to the right of the split point from the given word and its reversed word are generated. This results in 12 additional features that could be included in the feature vector set.

Thus, each split point of a word could have up to 18 features. Figures 2 and 3 help the reader visualize the features. Figure 2 shows various split points of a sample word "*aeroplane*" and its reverse. Figure 3 show various category of features for one chosen split point.

3.4 Implementation details

Our proof-of-concept framework was implemented in Python3. To facilitate memory efficient computation of subwords count, a trie data structure is utilized. A list of lists is used for representing features of each split point of a word. Random Forest from scikit-learn library is deployed for classification.



ae roplane, enalpor ea

Substring entropy	Substring count	Character N-grams
Word: H(a) : Pre H(ae): Current H(aer): Post	Word: Count(a): Pre Count(ae): Current Count(aer): Post	Word: (ae, ae, e, r, ro, rop)
Reverse word: H(enalpo) : Pre H(enalpor): Current H(enalpore): Post	Reverse word: Count(enalpo) : Pre Count(enalpor): Current Count(enalpore): Post	Reverse word: (por, po, r, e, ea, ea)

Figure 3: Features for a sample split point

4 Evaluation of Framework

4.1 Experimental Setup

The effectiveness of our framework is evaluated using DBLP titles and an initial vocabulary list is generated. Root words for ~2700 words are obtained from a knowledge base. The vocabulary list from DBLP dataset is appended to include the new words from knowledge base. This subset of ~2700 words serves as labelled data set for evaluating our framework. For each word in the new vocabulary list, a list of feature vectors for every potential split point is generated. If multiple root words exist, the longest root word is considered for label encoding. The classifier performance is evaluated using average accuracy across 10 iterations of randomly chosen train (80%) and test (20%) samples from the labelled data set. For accuracy calculations, subwords are generated per word as per the split points generated by the classifier. If any one of the subwords matches given roots of a word, then this word is counted towards true positive. The classifier is evaluated by plotting the accuracy for various subsets of features as shown in Figure 4. The threshold for probabilistic random forest at testing phase is set to 0.5.

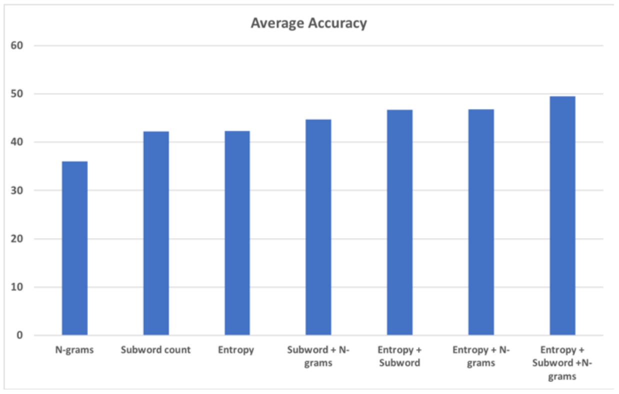


Figure 4: Accuracy across features

4.2 Analysis of results

The best average accuracy with threshold set to 0.5 is 49.5%. To understand ~50% failure rate, we divided the words in failed category into two categories: no-splits and wrong-splits. No-splits are words that have zero split points (and no subwords) and wrong-splits are words whose subwords do not match any given roots of a word. There were ~28.8% of words in no-split category and ~21.7% of words in wrong-split. To reduce the percentage of words in no-split category, the threshold of class probability is reduced, and a comparative study of threshold settings is shown in Table 1. It is evident from experimentation that reducing the class probability threshold reduces the no-split percentage and increases the average accuracy. The best average accuracy obtained in our framework with all features is 65.4% with class probability threshold set to 0.3. The reduction in threshold increases the wrong-split percentage as some no-split words are incorrectly split.

Test Set Parameters (%)	Thresh old @ 0.5	Threshol d @ 0.4	Threshol d @0.3
Accuracy	49.5	59.24	65.43
No-split	28.8	15.66	5.75
Wrong-split	21.7	25.10	28.82

Table 1: Study of Class Probability Threshold

4.3 Experiment Setup for Unsupervised Morpheme Extraction

The effectiveness of our classifier framework is evaluated by comparing it against the state-of-the-art unsupervised morpheme extraction techniques. Unigram language model [11] and BPE [10] from SentencePiece [21] are used for comparison. DBLP title data is used to build the unsupervised model and generate segments for every word. A subset of words that are present in labelled data set is used for accuracy calculation. For each word, segmentation is considered successful if one of the proposed subwords is present in the list of roots in the labelled data set. Both the models were tuned to output best performance by varying the vocabulary size. The performance of unsupervised framework is shown in Table 2.

Model Type	Vocabulary Size	Accuracy (%)
Unigram	8K	14.66
Unigram	16K	18.42
Unigram	19K	20.45
BPE	8K	14.59
BPE	16K	16.47
BPE	20K	17.50
BPE	24K	17.10
BPE	32K	16.25

Table 2: Unsupervised Morpheme Extraction Performance

4.4 Comparative Study

A comparison table of accuracy for supervised and unsupervised methods is shown in Figures 5 and 6. It is evident that even the lowest accuracy of the supervised framework (36%) is higher than the best accuracy of the unsupervised framework (20%) for DBLP title dataset. Additionally, the supervised framework with all features has ~2.5x better accuracy with class probability threshold set to 0.5. By reducing the class probability threshold, the overall improvement is ~3.2x, resulting in accuracy of ~65%. Table 3 shows various word splits across different methods.

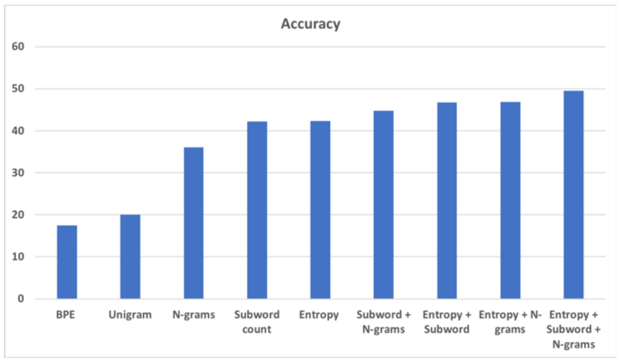


Figure 5: Accuracy of Supervised and Unsupervised methods

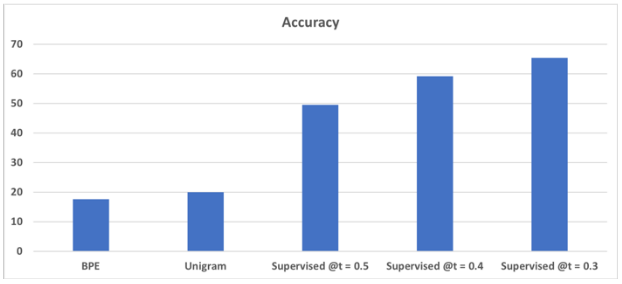


Figure 6: Accuracy with threshold variation

Word	Root	BPE	ULM	Supervised
inwards	in	in-wards	in-wards	in-wards
xylophone	xyl	xy-lo-phone	xy-lo-phone	xylo-phone
heterozygous	zyg	heter-o-zy-g-ous	hetero-zy-g-ous	hetero-zyg-ous
preponderance	ponder	pre-p-on-der-ance	pre-pon-der-ance	preponder-ance
carpology	carp	car-p-ology	car-p-ology	carp-ology
commemorate	com	comm-em-or-ate	co-mme-mo-rate	com-memorate

Table 3: Comparison of word splits across various methods

5 Future Improvements

One potential future enhancement is iterative classification, following principles from Segphrase [14]. A subset of words that have morphemes extracted at testing phase could be fed back into training set to help in better generalization of tree for all types of prefixes and suffixes. Another future improvement is based on further analysis of failure rate of the classifier. The main contributor for failure rate is under-splitting of compound words. One possible way to improve accuracy is to use a binary recursive classifier to re-split each subword until a minimum length is reached. However, this scheme requires substring transition entropy and substring count entropy for all substrings for a word, rather than just prefixes and suffixes.

6 Conclusion

We implemented a data-driven classification-based morpheme extraction framework. The framework included three categories of features: substring transition entropy, substring count and character n-grams to potentially identify a morpheme boundary. We evaluated our framework with DBLP title dataset and compared to existing state-of-the-art unsupervised morpheme extraction methods. The lowest accuracy from our framework (~36%) is better than the best accuracy from unsupervised techniques (~20%). The best accuracy from our framework is ~65%, a significant 225% improvement over unsupervised methods.

Acknowledgements

We sincerely thank Ahmed El-kishky, Ph. D student in the Data Mining group at UIUC for his excellent ideas, insights and guidance throughout the course of this project.

REFERENCES

[1] http://www.iflscience.com/technology/how-much-data-does-the-world-generate-every-minute/ Networks:Principles and Methodologies, Morgan & Claypool, pp.126, 2012

[2] https://www.ibm.com/blogs/insights-on-business/consumer-products/2-5-quintillion-bytes-of-data-created-every-day-how-does-cpg-retail-manage-it/

[3] Nadeau, David and Sekine, Satoshi. "A survey of named entity recognition and classification." Linguisticae Investigationes 30, no. 1 (2007): 3--26.

[4] Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL '10). Association for Computational Linguistics, Stroudsburg, PA, USA, 384-394.

[5] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In Proceedings of ACL, pages 1555–1565

[6] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI'16). AAAI Press 2741-2749

[7] Cícero Nogueira Dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32 (ICML'14), Eric P. Xing and Tony Jebara (Eds.), Vol. 32. JMLR.org II-1818-II-1826.

[8] Luong, Thang, Richard Socher and Christopher D. Manning. "Better Word Representations with Recursive Neural Networks for Morphology." CoNLL (2013).

[9] Cotterell, Ryan & Schütze, Hinrich & Eisner, Jason. (2016). Morphological Smoothing and Extrapolation of Word Embeddings. 1651-1660. 10.18653/v1/P16-1156.

[10] Sennrich, Rico & Haddow, Barry & Birch, Alexandra. (2016). Neural Machine Translation of Rare Words with Subword Units. 1715-1725. 10.18653/v1/P16-1162.

[11] Taku kudo. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates, ACL 2018

[12] Mathias Creutz, Krista Lagus, "Unsupervised discovery of morphemes", MPL '02 Proceedings of the ACL-02 workshop on Morphological and phonological learning - Volume 6, Pages 21-30

[13] Delphine Bernhard, "Unsupervised Morphological Segmentation Based on Segment Predictability and Word Segments Alignment", Proceedings of 2nd Pascal Challenges Workshop, 2006

[14] Jialu Liu, Jingbo Shang, Chi Wang, Xiang Ren, Jiawei Han, "Mining Quality Phrases from Massive Text Corpora", 2015 ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'15)

[15] Jingbo Shang, Jialu Liu, Meng Jiang, Xiang Ren, Clare R. Voss, Jiawei Han, "AutoPhrase: Automated Phrase Mining from Massive Text Corpora", ArXiv, Feb. 2017

[16] Ahmed El-Kishky, Yanglei Song, et. al. "Scalable Topical Phrase Mining from Text Corpora", Proceedings of the VLDB Endowment, Volume 8 Issue 3, November 2014, Pages 305-316

[17] Maria Danilevsky, Chi Wang, Nihit Desai, Xiang Ren, Jingyi Guo, and Jiawei Han. "Automatic Construction and Ranking of Topical Keyphrases on Collections of Short Documents ", SIAM Data Mining Conf. (SDM), 2014

[18] Stefan Schulz, Martin Honeck, Udo Hahn, "Biomedical text retrieval in languages with a complex morphology", BioMed '02 Proceedings of the ACL-02 workshop on Natural language processing in the biomedical domain - Volume 3, Pages 61-68.

[19] Fu, Guohong & Kit, Chunyu & Webster, Jonathan. (2008). Chinese word segmentation as morpheme-based lexical chunking. Inf. Sci.. 178. 2282-2296. 10.1016/j.ins.2008.01.001.

[20] Ahmed El-Kishky, Frank Xu, Aston Zhang, Stephen Macke and Jiawei Han, "Entropy-Based Subword Mining for Word Embeddings ", in Proc. of the 2nd Workshop on Subword and Character Level Models in NLP (SCLeM'18) (at NAACL 2018), New Orleans, LA, June 2018.

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