TOKENIZATION TECHNIQUES IN NLP:A COMPREHENSIVE REVIEW

Dr. Bhavesh M. Patel[1], M. Sule[2]

[1] Assistant Professor, Department of Computer Science, H.N.G.University, Patan, Gujarat, INDIA
[2] Scholar, Department of Computer Science, H.N.G.University, Patan, Gujarat, INDIA

Abstract

Tokenization is a natural language processing (NLP) preprocessing technique that involves breakingdown a text into individual tokens, often words or subwords. It has been the subject of extensive research in recent years, leading to the creation of numerous innovative methods and tools. The state of the art in NLP tokenization approaches is thoroughly reviewed in this study. We start by outlining the various tokenization techniques, including word, subword, and character-level tokenization. The benefits and drawbacks of various tokenization strategies, including rule-based, statistical, and neural network-based techniques, are then covered. The performance of various tokenization techniques and libraries is then compared.We also look at current developments in tokenization research,such as the use of unsupervised techniques and contextual data. Finally, we list many challenges in tokenization. We wrap up by examining prospective possibilities for tokenization research in the future and their effects on the larger discipline of NLP.

Keywords: NLP, tokenization strategies

Introduction

Natural language processing (NLP) has gained prominence in recent years with applications in fields like text categorization [1,2,3], sentiment analysis [4,5,6], machine translation [7,8], and information extraction [9,10]. Huge applications have been seen in recent times with the introduction of the transformer neural network architecture [11].

Akeystepin NaturalLanguageProcessing(NLP)istokenization[12],thisprocessisa keycomponentof NLP and is essential to many NLP applications [12].

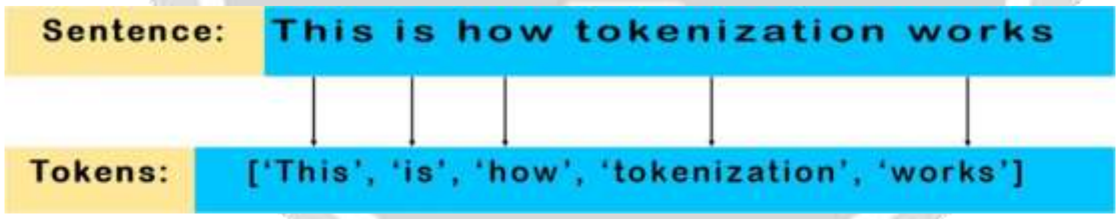

Sentence:
This is how tokenization works
Tokens:
['This', 'is', 'how', 'tokenization', 'works']

Figure 1: Tokenization

Fundamentally, tokenization is as simple as breaking a text into multiple chunks for further processing as seen in figure 1, but practically it is a difficult process because it necessitates an understanding of the linguistic structure of the text being processed [12]. There have been severaltokenizers proposedoverthe years, each havingits strengths and weaknesses. Traditionalrule-basedtokenizers have along historyand can sometimes be effective since they use manually written rules. Their inability to handle complex linguistic phenomena, like idiomatic expressions and compound words, led to the development of statistical and neural network-based tokenizers, whose popularity has significantly increased in recent years due to their capacity to handle a wide range of linguistic structures and nuances.

Word tokenizers, subword tokenizers, and character-level tokenizers are a few of the several kinds of tokenizers. While subword tokenizers break down words into smaller parts like prefixes and suffixes, word tokenizers try to divide a text into words. Individual characters are separated from a text by character-level tokenizers. The choice of tokenizer depends on the particular task and the features of the text being processed. Each form of tokenizer has benefits and drawbacks of its own.

1. Rule-BasedTokenizers

Traditionally,text was divided intotokens usingrules that aremanually created.Thismethodis known as rule-based tokenization. These guidelines, which are frequently based on heuristics like punctuation and whitespace, are made to deal with linguistic problems like compound words and abbreviations.

Rule-based tokenizers can handle enormous amounts of text data since they are typically straightforward and quick. They are also very adaptable because new rules can be added or updated to enhance their functionality for certain scenarios. But they do have some restrictions. First of all, they have trouble handling nuanced and complicated linguistic structures like idiomatic expressions and nested structures. Secondly,theyrely ontheexistenceofacompletesetofrules,whichcanbechallenging.Finally,noisyor sloppy text data may be difficult for rule-based tokenizers to manage.

Despite these challenges, they are still used in real-world projects and have been effectively deployed in numerous NLP applications. The Penn Treebank Tokenizer [13], the NLTK Tokenizer [14], and the Stanford Tokenizer [15] are a few popular rule-based tokenizers that are reviewed in this paper.

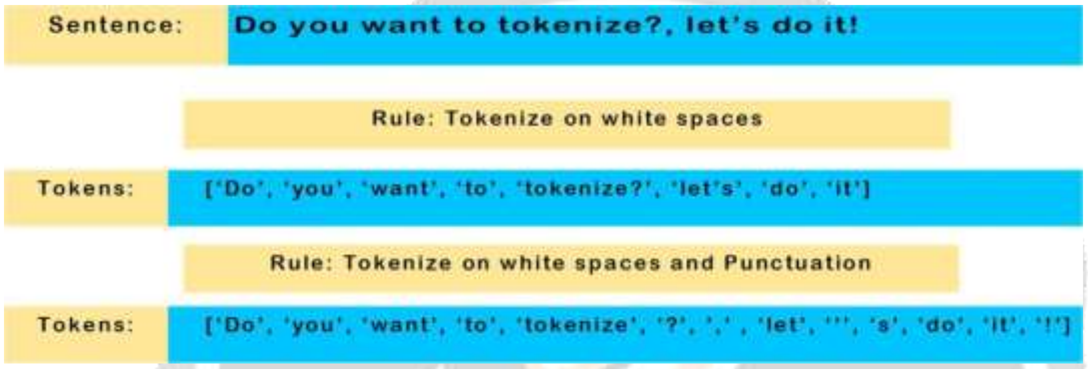

Sentence:
Do you want to tokenize?, let's do it!
Rule: Tokenize on white spaces
Tokens:
['Do', 'you', 'want', 'to', 'tokenize?', 'let's', 'do', 'it']
Rule: Tokenize on white spaces and Punctuation
Tokens:
['Do', 'you', 'want', 'to', 'tokenize', '?', ',' , 'let', ''', 's', 'do', 'it', '!']

Figure2:Rule-BasedTokenizer

Rule-based tokenizers have the benefit of being transparent since the rules are clear and concise, also it works very well with datasetsthat require a specific form of customizationlike tweets. Furthermore,rule- based tokenizers are frequently language-specific, making them more useful for dealing with languages that have certain properties.

However, they are typically not highly resistant to linguistic and textual variances. For instance, they could find it difficult to process text data that is loud, contains errors, or is written in a language otherthan the one for which they were intended. Furthermore, they might not be able to handle text material that comprises uncommon or uncommon linguistic structures.

Although rule-based tokenizers have been used widely in the past and are being used today, statistical or neural network-based tokenizers are more used today because they can handle a larger range of linguistic phenomena and are more adaptable to various text kinds and languages. However, rule-based tokenizers could still be helpful in some circumstances and might be a good choice for researchers and practitioners who have specific requirements or limitations for their NLP applications.

In the next section, we will discuss statistical tokenizers, which represent an alternative approach to tokenization that relies on statistical models to segment text into tokens.

1.1Examples ofrule-based tokenizers

Someexamplesofpopularrule-basedtokenizers include:

1. Penn Treebank Tokenizer [13]: This tokenizer was developed as part of the Penn Treebank project, which was a large-scale annotated corpus of English text. The Penn Treebank Tokenizer uses a set ofrules to segment text into tokens based on whitespace, punctuation, and other heuristics.

2. NLTK Tokenizer [14]: The Natural Language Toolkit (NLTK) is a popular Python library for NLP, and it includes a rule-based tokenizer that is highly customizable. The NLTK tokenizer includes rules for handling common linguistic phenomena, such as contractions, abbreviations, and hyphenated words. NLTK has

three different rule-based tokenization algorithms as TweetTokenizer for Twitter Tweets, MWET for Multi-word tokenization, along with TreeBankTokenizer for the English Language rules. Rule-based Tokenization helps perform the tokenization based on the best possible conditions for the nature of the textual data.

3. Stanford Tokenizer [15]: The Stanford Tokenizer is another popular rule-based tokenizer that is part of the StanfordCoreNLPsuite oftools. Thetokenizerusesa setofrulestosegment text into tokensbased on whitespace, punctuation, and other heuristics. It also includes rules for handling specific linguistic structures, such as email addresses and URLs.

4. OpenNLP Tokenizer: OpenNLP is an open-source library for NLP that includes a rule-based tokenizer. The OpenNLP Tokenizer uses a set of rules to segment text into tokens based on whitespace, punctuation, and other heuristics. It is highly customizable, allowing users to add or modify rules to improve its performance for specific use cases.

These are just a fewexamples of themany rule-based tokenizers that have been developed over the years. While these tokenizers can be effective in certain circumstances, they are generally less popular than statistical or neural network-based tokenizers, which can handle a wider range of linguistic phenomena and are more adaptable to different text types and languages.

2. Statistical Tokenizers

In contrast to rule-based tokenization [13,14,15], statistical tokenization employs machine learning algorithms to automatically identify patterns in text input and divide it into tokens. Statistic tokenization may learn from the data and adjust to differences in language and text type, unlike rule-based tokenization, which uses a predetermined set of rules to split text.

Maximum entropy modeling is one popular statistical tokenization method[16].A corpus of news articles or social media posts can be used to train maximum entropy models, which are probabilistic. The training procedure aims to discover a collection of parameters that minimizes the model's complexity while increasing the likelihood of the observed data.

A maximum entropy model [16] can be used to forecast the likelihood of various token boundaries in fresh text data once it has been trained. Using the context surrounding each prospective token boundary— for example, the characters before and after it—this prediction method determines which border has the highest probability before choosing it.

Conditional random fields are another statistical tokenization method (CRFs) [17]. A particular class of graphical model called a CRF [17] can be trained on labeled text data to discover patterns in the data and divide it into tokens. CRFs are probabilistic models [17], similar to maximum entropy models [16], that can adjust to changes in language and text type and learn from the data.

One benefit of statistical tokenization over rule-based tokenization is that it may be more resilient to fluctuations in language and text type. Statistical tokenizers are more versatile and adaptive to a wide range of NLP tasks because they can recognize patterns in the data and adjust to changes in language and text type.

Statistical tokenizers, however, can also require more computation than rule-based tokenizers, especially during the training stage. Additionally, because it can be challenging to comprehend why a specific tokenization was created, they may be more opaque than rule-based tokenizers.

2.1 Advantages and limitations of statistical tokenizers

Advantages:

1. Adaptability: Statistical tokenizers [14,15,20] are more versatile than rule-based tokenizers because they can adjust to various languages, text kinds, and writing styles. Statistical tokenizers [14,15,20] can automatically adapt to text variations and continuously improve their effectiveness by identifying trendsin the data.

2. Accuracy: When statistical tokenizers are trained on a lot of labeled data, they can reach high levelsof accuracy in tokenization. This precision can be very helpful in NLP applications like sentimentanalysis or named entity recognition that call for precise tokenization [21].

3. Robustness: Statisticaltokenizersarecapableof handlingavariety oflanguage phenomena[22],such as

contractions, abbreviations, and unidentified terms, which might be difficult for rule-based tokenizers. Statistical tokenizers may be better suited for processing text data in the actual world due to their robustness.

Limitations:

1. Statistical tokenizers need a lot of labeled training data to function properly [23]. For languages ortext typeswithout substantialannotatedcorpora,thismaybe a limitingfactor.Statistical tokenizersmight not be able to learn the patterns required for precise tokenization without enough training data [23].

2. Computer complexity: Statistical tokenizers might need a lot of computing, especially during the training stage [24,25]. They may be less appropriate for usage in real-time applications or on machines with little processing power as a result.

3. Interpretability: Rule-based tokenizers, which rely on explicit rules that are simple to inspect and comprehend, are more readable than statistical tokenizers [26], which can be harder to grasp. It can be challenging to comprehend why certain tokenization was created because the inner workings of statistical tokenizers can be more opaque.

2.2 Examples of statistical tokenizers

Here are some examples of statistical tokenizers:

1. NLTKTokenizer[14]:NLTK(Natural Language Toolkit)is a popular Python library for NLP, which includes a statistical tokenizer based on the Punkt algorithm. The Punkt algorithm uses unsupervised machine learning to learn patterns in text and automatically segment them into tokens.

2. Stanford Tokenizer [15]: The Stanford Tokenizer is another popular statistical tokenizer that uses a maximum entropy model to segment text into tokens. It is part of the Stanford CoreNLP suite, which includes a wide range of NLP tools and models.

3. Spacy Tokenizer: Spacy is a Python library for NLP that includes a statistical tokenizer based on a convolutional neural network (CNN) architecture. The Spacy tokenizer is optimized for speed and accuracy and can handle a wide range of text types and languages [18].

4. OpenNLPTokenizer:OpenNLPisaJava-based NLPlibrary that includes a statistical tokenizer based on maximum entropy modeling [19]. The OpenNLP tokenizer can be trained on custom data and used to segment text in a wide range of languages.

5. Apache Lucene Tokenizer [20]: The Apache Lucene Tokenizer is a statistical tokenizer that is part of the Apache Lucene search engine library. It uses a maximum entropy model to segment text into tokens and can be used for a wide range of NLP and search applications.

It's often crucial to assess a statistical tokenizer's performance on new data after it has been trained. A tokenizer's precision, recall, and F1 score are the three metrics that are most frequently used to assess it [27]. F1 score is the harmonic mean of precision and recall [28]. Precision measures the proportion of properly recognized tokens out of all identified tokens, recall is the proportion of correctly identified tokens out of all actual tokens.

A test set of annotated text is usually used to assess a statistical tokenizer. The test set is divided into two sections: a training set for the tokenizer and a development set for optimizing themodel and choosing the optimum parameters. When the model is trained

Other measures, such as accuracy, coverage, and speed, can be used to assess a tokenizer in addition to precision, recall, and F1 score [29,30]. While coverage represents the percentage of tokens that were properly identified out of all potential tokens, accuracy reflects the overall accuracy of the tokenization. For real-time applications, speed can be very crucial.

The Penn Treebank [13] and the CoNLL datasets [31,32,33,34,35] are two examples of benchmark datasets that can be used to assess tokenizers. These datasets are frequently used in NLP research [36,37] and offer a uniform approach to assessing the effectiveness of various tokenization models.

3 Neural Network-Based Tokenizers

A new method for text segmentation in NLP called neural network-based tokenization makes use of machine learning models, particularly neural networks, to recognize and divide the text into tokens. This method is distinct from statistical [14,15,20]and rule-based approaches [13,14,15] that utilize explicit rules or models to pinpoint word boundaries.

To learn the patterns and structures that define tokens, a model is trained using a huge corpus of text data in the neural network-based approach to tokenization. To accurately segment fresh text into tokens without the need for explicit rules, the model must first be trained using a significant quantity of training data and processing resources.

The capacity of neural network-based tokenization to handle changes and departures from conventional tokenization rules is one ofits advantages. The modelcan learn to recognize and segment words that may haveirregularspellingsormorphologicalvariants becauseit wastrainedonsuchabigcorpusoftextdata. It is crucial for languages with complex morphology to be able to accommodate such variances.

The most well-liked neural network-based tokenization method is SentencePiece [38]. SentencePiece is a library that offers a multilingual and multiscript general-purpose unsupervised text tokenizer and detokenizer.

Despite being a promising method for text segmentation in NLP, neural network-based tokenization has significant drawbacks and difficulties that need to be resolved. The need for a lot of training data and computer power to train the models is one of the main difficulties. A risk of overfitting exists as well when the model performs well on training data but poorly on fresh, untested data.

Despite these difficulties, the creation of tokenization models based on neural networks has the potential to greatly enhance the precision and resilience of text segmentation in NLP. Researchers may contributeto the advancement of NLP and open up a larger range of applications by continuing to create and improve these models.

3.1 Advantages and limitations of neural network-based tokenizers

When selecting the best tokenizer for an NLPassignment, it's crucial to take intoaccount the benefits and limits of neural network-based tokenizers.

Advantages:

1. Increased accuracy: In many NLP tasks, neural network-based tokenizers have been demonstrated to be more accurate than rule-based or statistical tokenizers [39]. This is so that the models can learn to recognize intricate structures and patterns that can be challenging to express explicitly.

2. Robustness to variations: Tokenizers based on neural networks are capable of handling changes and deviations from the norm. This is crucial for languages with intricate morphology and inconsistent spelling.

3. Flexibility: Neural network-based tokenizers can be trained on any form of text data, including news articles, scientific journals, and social media postings.

4. Neural network-based tokenizers can be customized to function in a variety of languages and industries. They may be trained on various datasets, which enables them to learn to adapt to various domains and capture language-specific properties.

Limitations:

1. The complexity of computation: Neural network-based tokenizers can be computationally expensive to train since they need a lot of training data and computing power.
2. Overfitting:Neural network-based tokenizers may perform poorly on new, unforeseen data if they have overfitted to the training data.
3. Lack of transparency:Because it can be challenging to understand how a modelgenerates predictions, neural network-based tokenizers are frequently referred to as "black boxes."
4. Limited interpretability:Because neural network-based tokenizers don't offer explicit rules or heuristics for text segmentation, it might be challenging to understand their outputs.

To determine the effectiveness of neural network-based tokenizers for various NLP tasks, it is critical to assess their performance.The evaluation of tokenizers built using neural networks may also be influenced by factors other than the regular metrics like Precision, Recall, Accuracy,etc.The quantity and caliber of the training data are two factors. While poor-quality training data can lead to overfitting and poor performance on new data, large and diverse training data can enhance the performance of neural network-based tokenizers.

The selection of hyperparameters, including the learning rate, batch size, and number of layers in the model, is another aspect. These hyperparameters must be tweaked properly for each NLP task because they have a major impact on the model's performance.

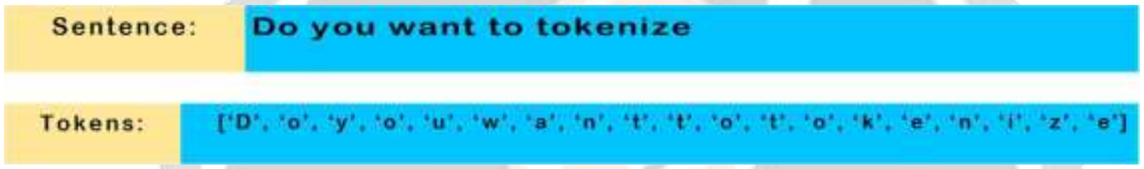
Researchers frequently employ benchmark datasets and accepted assessment frameworks, such as the CoNLL-2000 and CoNLL-2002 shared tasks, which offer accepted test sets and accepted evaluation criteria, to assess neural network-based tokenizers. To assess the relative efficiency of severaltokenization techniques, including rule-based and statistical tokenizers, researchers frequently compare the performance of neural network-based tokenizers to those methods.

4 Types of Tokenizers

There are various forms oftokenization, each with unique benefits and restrictions. Themost widely used and accepted ones such as character-level tokenizers, word-level tokenizers, sub-word-level tokenizers as well as Syllable-based and hybrid tokenizers will be reviewed in this paper.

a. Character-level tokenization

Tokenizingtextatthecharacterlevel,orcharacter-leveltokenization,involvestreatingeachcharacterasa separate token. Character-level tokenization occurs at the most fundamental level of text analysis, in contrast to word and subword tokenization, which separates text into discrete pieces based on semantic significance.



Sentence: Do you want to tokenize

Tokens: ['D', 'o', 'y', 'o', 'u', 'w', 'a', 'n', 't', 't', 'o', 't', 'o', 'k', 'e', 'n', 'i', 'z', 'e']

Character-level tokenization has the advantage of being able to handle uncommon and out-of-vocabulary words (OOV), which are sometimes problematic in word-based techniques. The reason for this is that character-level tokenization, as opposed to just relying on pre-defined word embeddings, can break down words into their component pieces, enabling the model to learn patterns and correlations between specific characters.

In situations when words are not clearly defined or separated, such as in Chinese or Japanese language, which is written without spaces between words, character-level tokenization can be helpful.

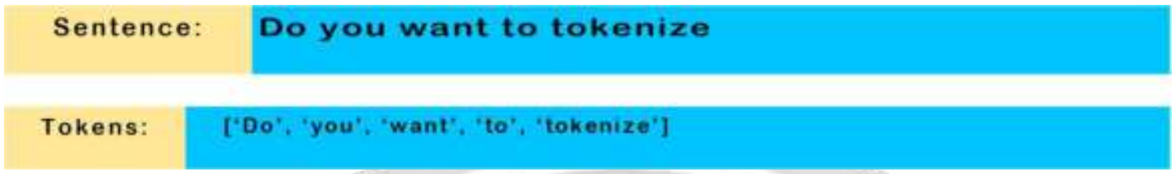
Character-level tokenization does have some disadvantages, though. It may be unable to understand the semantics of lengthier phrases or sentences because it only works at the character level and may have trouble understanding languages with complicated morphology.

Character-level tokenization has been effectively applied in several NLP applications recently, including sentiment analysis, machine translation, and text categorization. For instance, it has been demonstrated that character-level models are successful in predicting the sentiment of brief text communications, such as tweets or text messages, when conventional word-based models may fail because of the context's brevity.

Character-level tokenization is a helpful NLP technique that can aid in overcoming issues with conventional word-based techniques. Character-level tokenization can enhance the performance of NLP modelsbytreatingeachcharacterasaseparatetoken,especiallyinsituationswhenwordsarenotclearly defined or separated. However, it's crucial to be aware of the character-level tokenization's restrictionsand to select the best tokenization technique based on the particulars of the task at hand.

b. Wordtokenization

With this kind of tokenization,a text is broken up into individual words, frequently separated by spaces or punctuation. Many NLP applications, including sentiment analysis [1,23], text categorization [4,5,6], and language modeling, make use of word-based tokenization. Word-based tokenization's simplicity and ease of use are two of its key benefits. This strategy, nevertheless, may have limitations in languages like Chinese or Japanese where punctuation marks or spaces are not used to divide words.



Sentence: Do you want to tokenize

Tokens: ['Do', 'you', 'want', 'to', 'tokenize']

There are various techniques for word tokenization, from straightforward heuristics to more intricate machine-learning algorithms. Splitting the text on whitespace characters, such as spaces and tabs, is one of the simplest and most popular techniques. For many languages, including English, where words are generally separated by spaces, this strategy works effectively. However, it may be less effective for languages like Chinese or Japanese where words are not separated by whitespace.

Another popular method for word tokenization is regular expressions, which are patterns. Using regular expressions, which are patterns that can match particular patterns of letters in a text, is another popular technique for word tokenization. A regular expression, for instance, may match any string of characters except whitespace and punctuation. Although it may be more efficient than simple whitespace-based tokenization, this method calls for a deeper understanding of the language being processed and may be more challenging to put into practice.

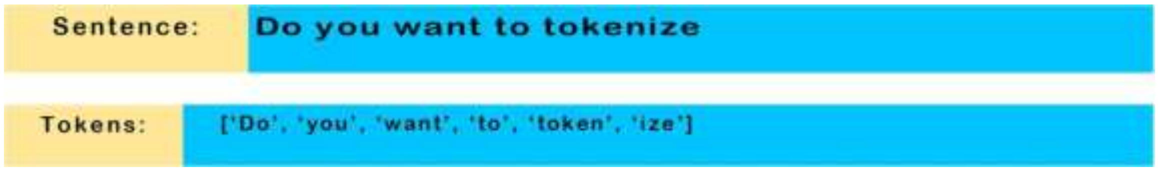
Word tokenization can also be done using machine learning algorithms like conditional random fields and support vector machines [40]. To predict the appropriate boundaries between words in a fresh text, these algorithms learn from samples of successfully tokenized text. Tokenization techniques based on machine learning may be more accurate than those that use rules, but they call for more training data and processing power.

Despite the various strategies for word tokenization, all techniques face many difficulties. Dealing with circumstances when it is unclear whether two words should be regarded as distinct tokens or as a single compound word poses one of the biggest hurdles. For instance, in English, "ice cream" might be regarded as either a single compound word or two independent words. Managing instances of grammatical errors or non-standard language, such as slang or misspelled words, in the text is another difficulty.

To sum up, word tokenization is a crucial step in many NLP applications, and there are a variety of techniques for carrying it out. Researchers and practitioners in NLP can choose the optimal strategy for their particular application and increase the accuracy and efficacy of their models by being aware of the benefits and drawbacks of each approach.

c. Subwordtokenization

A sort of tokenization called subword tokenization breaks words down into simpler components like syllables or character n-grams [38]. This strategy is especially helpful for morphologically complicated languageswherewordscantakeonnumerousinflectionsandderivations.Subwordtokenizationcanhelp a model learn fewer unique words and increase the precision of NLP models by breaking words into smaller parts [38].



Sentence: Do you want to tokenize

Tokens: ['Do', 'you', 'want', 'to', 'token', 'ize']

For subword tokenization, there are many techniques, such as byte pair encoding (BPE), unigram language modeling [41], and recurrent neural networks [42]. One of the most popular techniques for subword tokenization is BPE, a form of data compression algorithm that functions by substituting a single token for the most frequent pairings of bytes. Numerous NLP tasks, such as named entity identification, sentiment analysis, and machine translation, have seen success when using BPE.

WordPiece[43] is a well-known subword tokenization method that can be used with numerous other NLP models. To tokenize a given text into word pieces, WordPiece first pre-tokenizes the text into words by splitting on punctuation and whitespaces. Another technique for subword tokenization is known as unigram language modeling, which is based on teaching a language model to predict the next token given a sequence of tokens[43].The likelihood of each component given the context is then utilized to segment words into smaller subunits using the language model. It has been demonstrated that unigram language modeling works well for languages with complicated morphology, including Finnish and Turkish.

The tokenization of subwords can also be done using recurrent neural networks (RNNs) [42]. In this method, the subunits are concatenated to create the final word after the RNN is trained to predict the subsequent subunit given the prior subunits. It has been demonstrated that RNN-based subword tokenization works well for languages with complex morphology and can also be applied to languages with non-standard orthographies, like Chinese.

Subword tokenization is superior to word-based tokenization in many ways, including the ability to handle languages with complicated morphology,the ability to minimize model vocabulary, and the ability to increase the precision of NLP models. Subword tokenization does have some drawbacks, though, including a potential loss of interpretability and a rise in computing complexity.

Subword tokenization is a crucial NLP approach that can enhance the efficacy and accuracy of models for languages with complicated morphology. Understanding the various subword tokenization techniques allows NLP researchers and practitioners to select the optimal strategy for their particular application and enhance the performance of their models.

d. Syllable-based tokenization

Syllable-based tokenization is a technique for breaking down words into smaller parts based on their syllable counts is called. In languages like Korean or Japanese, where words are made up of several syllables, this approach is frequently employed. The accuracy of tasks involving natural language processing, such as speech recognition, machine translation, and text-to-speech conversion, can be increased by using syllable-based tokenization.

Syllable-based tokenization involves breaking words into individual syllables and treating each one as a separate token. For example, the word "telephone" would be broken down into two syllables, "tel" and "ephone". In some cases, a single syllable may be treated as a separate token, such as in the word "a".

One of the advantages of syllable-based tokenization is that it can improve the accuracy of natural language processing tasks in languages where word boundaries are not clearly defined. By separating words into their syllables, it becomes simpler to determine the meaning of the word because each syllable's meaning may be taken into account separately. In languages where words might have numerous meanings depending on the context in which they are used, syllable-based tokenization can also be helpful.

Syllable-based tokenization presents many difficulties. Choosing the proper syllable boundaries can be difficult because various speakers may pronounce the same word differently. Additionally, since certain languages do not use a syllable-based system for word construction, syllable-based tokenization may not be appropriate for all languages.

Syllable-based tokenization is still a crucial method in natural language processing despite these difficulties. Syllable boundaries are still being precisely identified, and new approaches of incorporate syllable-based data into tasks involving natural language processing are still being investigated.

e. Hybrid tokenization

Hybrid tokenization employs two or more of the methods listed above to enhance performance for a specific NLP task. For example, a hybrid method might use character-based tokenization for some words or phrases that are difficult to tokenize using a word-based technique alone before switching back to word-based tokenization for the rest of the text. To better balance recall and precision, it integrates many tokenization algorithms. A rule-based tokenizer and a statistical or neural network-based tokenizer are generally combined as part of the hybrid approach.

The hybrid approach's ability to handle the shortcomings of distinct tokenization techniques is one of its benefits. For instance, statistical or neural network-based tokenizers may yield too many false positives or have difficulty with unusual terms, whereas rule-based tokenizers may be overly basic and miss subtle nuances. Hybrid tokenizers can produce a more accurate and thorough examination of text by combining these strategies.

In a hybrid approach, tokenization approaches can be combined in a variety of ways. The text can be cleaned up and removed from noise using a rule-based tokenizer as a pre-processing step before beingsent to a statistical or neural network-based tokenizer for additional analysis. A different strategy is to employ several tokenizers concurrently and combine the outputs to produce a final set of tokens.

Machine translation, named entity recognition, sentiment analysis, and other applications have all used hybrid tokenization. However, creating a hybrid tokenizer that works well can be difficult because it necessitates carefully weighing the advantages and disadvantages of various tokenization methods and how they can be integrated.

f. Comparison of different tokenization types

This section of the paper will compare the different types of tokenization that we have discussed so far: word, subword, and character-level tokenization.

The most popular approach for NLP tasks is word tokenization, which has long been accepted as the norm. It works well in a variety of settings and with many different languages. Word tokenization, however, has restrictions when it comes to managing uncommon and out-of-vocabulary words, which might be problematic for some jobs and languages.

The shortcomings of word tokenization have been effectively addressed by subword tokenization. Subword tokenization can handle OOV terms and unusual words by dividing words into smaller units and can be used with a larger variety of languages. For jobs like machine translation, where the model must handle many languages and unidentified terms, subword tokenization is especially successful.

A more modern method that treats each character as a separate token is character-level tokenization. Itcan be used with OOV words, uncommon terms, and languages like Chinese or Japanese that do not utilize spaces to separate words. It might, however, have trouble capturing the semantics of lengthy expressions or sentences, and it might not work well for languages with intricate morphology.

Word-level tokenization is typically the most effective in terms of computational efficiency because it minimizes the size of the input sequence. Tokenization at the subword and character levels might lead to longer sequences and demand more computing power.

The particular needs of the work at hand and the characteristics of the text data determine which tokenization method should be used. Subword or character-level tokenization may be recommended for situations where OOV words or uncommon terms provide a challenge. Word-level tokenization may be themostadvantageouschoiceforjobswhencomputingperformanceisaconsideration.Thefinaldecision on the tokenization approach should be made after carefully weighing the trade-offs between accuracy, efficiency, and other aspects pertinent to the particular task.

TokenizationType	**Accuracy**	**LanguageSupport**	**Speed**	**Simplicity**
Character-level	Low	Supportsalllanguages	Fast	Simple
Word-level	High	Limitedlanguage support	Slow	Complex

Subword-level	Medium	Supportsmostlanguages	Medium	Complex
Syllable-based	Medium	Limitedlanguage support	Medium	Simple
Hybrid	High	Supportsmostlanguages	Fast	Complex

Table1:Generalized performance of tokenizers

g. Tradeoffs between the types of tokenizers

The various tokenization types all have their strengths and weakness, it is, therefore, important to understand the tradeoffs involved when making a choice.

In terms of Accuracy, because word-level tokenization differentiates tokens based on real language, it offers improved accuracy. Sub-word level tokenization also offers good accuracy. On the other hand, character-level tokenization and syllable-based tokenization are less accurate since they divide words into individual characters or syllables.word-level and sub-word-level tokenizers support a variety of languages. Character-level tokenization and syllable-based tokenization are somehow more robust than the others because they don't have language dependency issues.

In terms of speed, character-level tokenization, is the fastest. Syllable-based tokenization divides words into syllables according to language-specific norms, making it slower than character-level tokenization, but quicker than word-level and sub-word-level tokenization. Tokenization at the word and sub-word levels are the slowest since it involves complex language-specific processing.

In terms of simplicity, character-level tokenization is the most straightforward. The reason is that it doesn't call for any linguistic expertise.Syllable-basedtokenizationisalsoratherstraightforwardbecause itusеslinguisticnormstodividewordsintosyllables.Word-levelandsub-word-leveltokenizationare more difficult since they need processing that is specific to the language and an understanding of linguistics.

Character-level tokenization often produces a wider vocabulary and can handle out-of-vocabulary words better than other approaches, but because of the higher number of tokens involved, it can be slower and less accurate.

In terms of language support,Word-level tokenization is precise,although it can be language-specific and may struggle with uncommon or uncommon words. Subword-level tokenization creates a balance between character and word-level tokenization, leading to a smaller vocabulary that can accept words that aren't in the dictionary, but it can also be slower and less accurate than word-level tokenization. Syllable- based tokenization is helpful for languages that lack spaces between words or have complex spelling, although it can be slower and less precise than other approaches. Hybrid tokenization can incorporate the benefits of many approaches, but it might be more difficult to implement and might need more computing power. The task at hand, the language being utilized, and the data being used ultimately determine which tokenization technique should be employed.

5 Tokenization Tools and Libraries

Various tools make tokenization easy and simple to implement. These are widely accepted in the NLP community and have been used in a robust application

1. NLTK (Natural Language Toolkit) [14]: NLTK is a popular Python library for NLP tasks, including tokenization. It provides several tokenization methods, including word tokenization, sentence tokenization, and regular expression tokenization.
2. SpaCy: SpaCy is another popular Python library for NLP tasks, with a strong focus on efficiency and usability. It provides high-quality tokenization for a wide range of languages, including subword tokenization, which can handle OOV words and rare words.
3. StanfordCoreNLP[15]:StanfordCoreNLP is a Java-based toolkit that provides arange of NLPtools, including tokenization, sentence segmentation, and part-of-speech tagging. It is widely used in research and industry, particularly for tasks such as named entity recognition and sentiment analysis.
4. OpenNLP: OpenNLP is an Apache project that provides arange of NLPtools, including tokenization, sentence segmentation, and part-of-speech tagging. It is designed for performance and can handle large datasets.
5. Gensim [44]: Gensim is a Python library for topic modeling and vector space modeling. It includes

several tokenization methods, including word tokenization and regular expression tokenization.

6. TensorFlow [45]: TensorFlow is a popular deep learning framework that includes tokenization as a pre-processing step in several of its NLP models, including the transformer-based models used for language translation.

7. HuggingFace[46] Tokinizers: HuggingFace is a Python library that provides pre-trained models for a wide range of NLP tasks, including tokenization. Its transformers module includes several pre-trained models, such as BERT and GPT-2, that can perform tokenization as part of their processing pipeline.

8. Keras [47]: Keras is a high-level neural network library for Python that includes several pre-processing tools, including tokenization. It provides several built-in tokenization methods, including word-level, character-level, and subword-level tokenization.

9. NLTK Punkt [14]: The Punkt algorithm is a data-driven sentence segmentation algorithm included in the NLTK library. It is based on unsupervised learning and can be trained on specific domainsto improve its accuracy.

10. Moses: Moses is a statistical machine translation system that includes several pre-processing tools, including tokenization. It provides several tokenization methods, including word-level and subword-level tokenization, as well as tools for handling non-standard characters and symbols.

11. Stanford Word Segmenter: The Stanford Word Segmenter is a Java-based tool that provides several tokenization methods, including word-level and subword-level tokenization. It can handle multiple languages, and its output can be used as input for other NLP tools, such as part-of-speech taggers and named entity recognizers.

When it comes to tokenization, there is no universally applicable approach. The most appropriate tokenizer will vary depending on the particulars of the task at hand and the properties of the data being processed. The above libraries provide a range of tokenization methods, from traditional rule-based methods to more modern neural network-based methods. They are also multilingual and can generalize well with a lot of languages.

6 Recent Advances in Tokenization Research

Recent developments in tokenization research have concentrated on enhancing the precision and effectiveness of tokenization as well as investigating novel tokenization strategies.

Enhancing tokenization's precision has been the subject of research, especially for languages withintricate scripts and morphologies. Convolutional neural networks (CNNs) [48] and recurrent neural networks (RNNs) [42], for instance, have been studied as deep learning models for tokenization. These models have been used to increasethe precision of tokenization for English andother languages,and they have demonstrated promising results for tokenizing languages with complex scripts, such as Arabic.

The effectiveness of tokenization has also been the subject of investigation. This involves accelerating tokenization on multi-core systems or distributed computing environments by using parallel processing techniques, as well as by creating quicker and more memory-efficient algorithms.

Recent studies have also investigated novel tokenization techniques, including simultaneous tokenization and segmentation and unsupervised tokenization. Without the aid of pre-established rules or training data, unsupervised tokenization involves learning token boundaries from unannotated text input. Combining tokenization and segmentation into a single process, known as joint tokenization and segmentation, has produced promising results for languages with intricate scripts and morphologies.

The integration of tokenization with other NLP tasks, like named entity recognition and part-of-speech tagging, has also been studied recently. Recent developments in tokenization research have demonstrated encouraging outcomes for enhancing the precision and effectiveness of tokenization as well as investigating novel tokenization methods.

6.1 Future Impact of Tokenizerson NLP Systems and Applications

i. Contextual information in tokenization

To establish the boundaries of tokens in a text, contextual information in tokenization refers to the utilization of nearby words or phrases. Since the emergence of deep learning models [42,45] that can incorporate contextual information into their predictions, this method of tokenization has grown in popularity. Using a language model to anticipate the most likely token boundaries in a given text is a typical example of contextual information in tokenization. Language models can be used to forecast the likelihood that a specific string of words or characters will appear in a given context after being trained on a lot of text data. Language models can increase the accuracy of tokenization by adding this contextual information, especially for languages with complicated scripts and morphologies.

To help establish token boundaries, part-of-speech (POS) tagging [49] is another method of contextual tokenization. POS tagging includes identifying the part of speech for each word in a text(e.g.,noun,verb, adjective). The limits of multi-word expressions, such as compound nouns or phrasal verbs, can be determined using this information since they might not be obvious based solely on surface-level characteristics.

Additionally, uncertain conditions in tokenization can be handled using contextual information. The apostrophe can be used, for instance, in English to denote contractions like "don't" or possessive forms like "John's automobile." Surface-level features alone may not always be sufficient to decide whether the interpretation is accurate. A tokenizer can more precisely determine the right token boundaries by taking contextual information, such as the surrounding words or the context of the sentence, into account.

ii. Unsupervised tokenization methods

Unsupervised tokenization approaches establish token boundaries without using labeled training data. These techniques are especially helpful for languages or text types where there are few or no labeled data sources.

Unsupervised word segmentation, which involves determining word boundaries based on statistical patterns in the text, is a well-liked unsupervised tokenization technique. Languages like Chinese [50] or Japanese, which have few or no spaces between words, frequently employ this technique. Based on the frequency and distribution of a character sequence in the text, unsupervised word segmentationalgorithms estimate the chance that a given character sequence is a word.

Byte pair encoding (BPE), another unsupervised tokenization technique, works by repeatedly joining the most frequent character pairs in a text to produce new tokens. BPE is frequently used in neural machine translation and other sequence-to-sequence tasks because it may divide the text into smaller pieces thatare simpler for neural models to process.

Unsupervised tokenization techniques can be quite successful, especially when labeled data is hard to get by or in short supply. These approaches might not always be as precise as supervised approaches that use labeled training data. To obtain high levels of accuracy, unsupervised approaches may also be more computationally demanding and call for bigger amounts of text data.

In general, unsupervised tokenization techniques offer a helpful substitute for conventional supervised techniques, especially for languages or text types for which there is a dearth of labeled data.Unsupervised approaches are likely to play a bigger role in NLP research and applications as huge, unlabeled text corpora become more widely available.

iii. Other recent advances

There have been many recent innovations in tokenization research in addition to the improvements in supervised and unsupervised tokenization techniques. The adoption of neural network-based models for tokenization is one noteworthy development. Since neural network-based models can learn intricate textual patterns and adapt to many languages and text kinds, they can be quite effective for tokenization tasks. For Chinese word segmentation, researchers have created neural network-based models that perform better than conventional rule-based and statistical approaches.

The incorporation of contextual data into tokenization models is another recent development in tokenization research. In complicated or confusing language, contextual information, such as nearby words or sentences, might assist in determining the borders of tokens. Recent research, for instance, has looked at how contextual information might be used in tokenization models for social media material, which can be quite casual and utilize uncommon spellings and syntax.

The use of tokenization for tasks other than text processing, such as speech recognition and image captioning, has also been gaining popularity. For instance, using tokens to describe visual concepts and spatial relationships, researchers have looked into the potential of tokenization for automatically creating descriptions for photos. Recent developments in tokenization research have resulted in the creation ofnew techniques and tools that can analyze text and other types of data more precisely and effectively. These developments are anticipated to become more crucial in a variety of applications and domains as the field of NLP continues to expand and change.

7.Challenges in Tokenization

Although tokenization research has recently made strides, there are still many issues that need to be resolved before the accuracy and efficiency of tokenization tools and techniques can be increased.

Handling words that are not often used in context (OOV) is one of the main issues in tokenization. OOV words are words that are not part of a predefined lexicon or dictionary. Because they may be misspelledor have several potential tokenizations, OOV words can be particularly challenging to manage during tokenization. Incorporating techniques from unsupervised or semi-supervised learning can help tokenization models adjust to new words and text kinds, which is one way to deal with this problem.

Handling text that incorporates non-standard or informal language, such as slang, abbreviations, or misspellings, presents another issue. This is especially important in online platforms like social media where people may use informal language to express themselves. Tokenization models must be able to manage differences in language and spelling while still precisely recognizing and isolating individual tokens to meet this issue.

Making sure that tokenization models can handle multilingual material, which may include various writing systems, grammatical structures, and tokenization conventions is another problem. Multiple writing systems and languages can be handled by models created by researchers, however, these models can be complicated and require a lot of training data.

Finally, a thorough examination of tokenization methods is required, especially concerning human annotators. While there are many well-established measures for measuring tokenization accuracy, additional standardized assessment datasets and processes are still required, along with more effective techniques for testing models on more difficult or subtle tokenization tasks.

7.1 Noisy and informal text

Noisy and informal text is defined as writing that is more conversational or casual and may contain slang, colloquialisms, abbreviations, and typos. Because it might not adhere to accepted grammatical standards or contain words that aren't frequently found in official dictionaries, this kind of content might be extremely difficult to tokenize.

Researchers have looked into a variety of tokenization techniques to manage noisy and informal writing. Tokenizationmodelscanbemodifiedtofitaparticularlanguageortypeoftextusingunsupervisedor

semi-supervised learning techniques. To manage differences in spelling and word structure, character- level tokenization or subword tokenization may be used.

Another strategy is to recognize and categorize tokens with the use of external knowledge sources, like dictionaries or domain-specific lexicons. To improve its accuracy, a tokenizer for social media writing,for instance, might employ a dictionary of widely used slang and acronyms.

The use of neural network-based models for processing noisy and informal text has also been investigated recently. These models can gradually adjust to new words and text formats and learn to recognize patterns and variances in language use.

Despite these developments, processing noisy and informal writings till presents difficulties. Misspellings and spelling variations, for instance, might be particularly challenging to manage because they may differ from what is found in conventional dictionaries or lexicons. Furthermore, the use of slang and informal language might change significantly among situations and societies, making it challenging to provide a universally applicable solution.

Trade-offs between speed and accuracy

The trade-off between accuracy and speed is a crucial one in tokenization. More intricate tokenization techniques are typically more accurate but also more computationally expensive, which can prolong processing times. Simpler tokenization techniques, on the other hand, are frequently quicker, but they might not be able to handle specific text kinds or they might produce more mistakes.

Rule-based tokenization techniques are frequently the quickest since they use regular expressions or straightforward heuristics to identify tokens. These techniques might, however, have trouble understanding morecomplicatedmaterial,such asthatwith erroneous grammar or peculiar abbreviations.

Since statistical and neural network-based approaches may learn to recognize patterns in the text and adjust to changes inlanguage use,they are typically more accurate than rule-based approaches. However, because they demand a lot of processing power to run and train, they may be substantially slower.

Utilizing a hybrid strategy that combines rule-based, statistical, and neural network-based methods is one way to strike a compromise between speed and accuracy. For instance, a tokenizer might utilize a straightforward rule-based approach for the majority of tokens but fall back on a trickier approach for tokens that it is unsure of.

Using pre-trained tokenization models is an alternative strategy that can be quicker while still producing high-accuracy results than training a model from scratch. Pre-trained models can be enhanced to perform better on particular text or language types.

The final decision on the tokenization approach will be made in light of the demands and limitations of a particular application. A more complicated approach—even if it is slower—might be required for jobsthat call for high accuracies, such as sentiment analysis or machine translation. On the other hand, even if it is less precise, a simpler approach might be better suited for jobs that demand quick processing times, such as real-time chat apps.

7.2 Other challenges

In addition to the challenges of noisy and informal text and the trade-offs between speed and accuracy, there are many other challenges in tokenization that researchers are currently working to address.

The problem of out-of-vocabulary (OOV) words is one of the main difficulties. OOV terms, which are words that are absent from a tokenization model's training set, can be problematic for statistical andneural network-based approaches that rely on pre-defined vocabularies. OOV words may be used with neworuncommonwords,propernouns,ortechnicalterminology,amongothercircumstances. Theuse of character-level or subword-level representations to enable the model to recognize partial matches or the expansion of the vocabulary using external knowledge sources like dictionaries or encyclopedias aresome of the approaches being developed by researchers to handle OOV terms.

The issue of multilingual and cross-lingual tokenization presents another difficulty. Because different languages have varied tokenization rules and norms, tokenizing text in many languages or across linguistic barriers can be challenging. Tokenization models that can accommodate various languages or even code-switching inside the same text are currently being developed by researchers. These models often rely on a combination of statistical or neural network-based models that can identify trends across languages and language-specific rules.

The difficulty of tokenizing text in particular fields or genres is the last. Scientific publications, social media posts, and legal documents all have various tokenization norms and needs. To tackle the unique issues of each domain, such as the use of technical terminology, abbreviations, or slang, researchers are striving to create domain-specific tokenization models.

8. Future of NLP Tokenization Research

Tokenization is still a crucial and active research issue as the science of natural language processing develops. A core NLP activity is tokenization, which is essential for many downstream applications including text categorization, sentiment analysis, machine translation, and others.

To significantly increase the accuracy of the current tokenizers is one of the main research objectives in tokenization. Tokenization methods need to be able to handle a variety of text data accurately due to the increasing demand for more complicated and specialized text processing. To increase the accuracy of tokenization, recent studies have investigated the use of deep learning models like transformers and unsupervised learning methods like clustering algorithms.

The creation of tokenization methods that can handle multilingual text data is a significant field of research.

Multilingual text processing is becoming more and more necessary, particularly for applications like machine translation and cross-lingual information retrieval.

Additionally, tackling the difficulties of noisy and informal writing, such as posts on social media and user-generated content, is another area of tokenization research. To do this, robust and flexible tokenization techniques must be created, capable of handling slang, typos, and other non-standard forms of language.

In tokenization research, speed and accuracy trade-offs are other crucial factors. Tokenizers must be able to processa lot oftext data quickly while still keeping highaccuracy standards. This necessitates creating efficient and effective tokenization algorithms, frequently through investigating novel tokenization paradigms.

In general, NLP tokenization research will keep concentrating on enhancing the precision, effectiveness, and flexibility of tokenization techniques to suit the requirements of various text processing applications. The creation of novel methods and tools that can manage the complexity of contemporary text data and get around the difficulties presented by the constantly changing linguistic environment will probably be required for this.

8.1 Potential directions for future research

As NLP continues to expand, there are still many areas that could benefit from further research on tokenization. Here are some potential directions for future research:

1. Improving accuracy in noisy and informal text: As was already established, noisy and informal writing can make accurate tokenization difficult. Future studies could concentrate on creating more reliable models that can handle such language better, perhaps by adding additional information to the tokenization procedure.

2. Developing better subwordtokenization models: There is still a lot that can be done to make subword tokenization even more effective at managing words that are not commonly used. Future studies could look into improved methods for creating subword units that are better suitable for various languages or text types.

3. Exploring unsupervised and semi-supervised tokenization methods: New tokenization models thatcan learn from unannotated or partially annotated data may be able to be created as unsupervised and semi-supervised learning techniques continue to gain prominence.

4. Investigating the impact of tokenization on downstream tasks: There is still a lot to learn about how various tokenization models can affect the performance of these tasks, even though tokenization is frequently viewed as a preprocessing step for subsequent NLP tasks. Future studies might examine how various tokenization techniques affect processes like machine translation, sentiment analysis, and question-answering.

5. Investigating tokenization for new types of data: There may be chances to create new tokenization models thatcanbettermanagethesedatatypesasnewtypesofdata,suchasspeech,andmultimodal data, become available for NLP.

6. Investigating trade-offs between speed and accuracy: Tokenization models that can strike a compromise between speed and accuracy may become more important as NLP's use in business applications continues to increase. The development of models that can tokenize quickly without compromising accuracy could be the main goal of future studies.

In conclusion, there are many potential areas for future research in NLP tokenization. As the field continues to grow and evolve, we can expect to see continued advances in this important area of NLP research.

8.2 Implications for the broader field of NLP

Tokenization is a fundamental task in Natural Language Processing (NLP), a topic that has advanced significantly in recent years. Tokenization's further development has ramifications for NLP as a whole.

The first is that NLP activities like part-of-speech tagging, parsing, and machine translation can all be directly impacted by the accuracy of tokenization. Thus, more accurate tokenization will result in more precise and effective NLP systems.

Second, there have been improvements in performance and new applications as a result of the growinguse of neural network-based tokenization models. These developments have opened up new possibilities for raising the bar for additional NLP jobs. For instance, a particular sort of neural network design called transformers has shown outstanding results in language modeling applications.

Thirdly, tokenization has been demonstrated to be a crucial element of AI that is understandable. The development of tools that enable end-users to comprehend how the models generate decisions has become necessary in response to the growing demand for more transparent and interpretable AI systems. Since tokenization is the foundation of the majority of NLP models, researchers have been looking into how to include tokenization information in the interpretability frameworks of these models.

Tokenization is still a crucial NLP activity that has a big impact on applications later on. Future breakthroughs inNLP are likely to benefit greatly from the growing amount of researchand development in tokenization, including recent developments in neural network-based models, unsupervised learning, and contextual information inclusion.

9. Conclusion

This thorough overview concludes by highlighting the important developments in tokenization research over the past several years, such as the creation of more advanced models and methods for handling noisy and informal text. The assessment also finds that while tokenization has many difficulties, it has a bright future ahead of it, and academics may anticipate continuous advancements in accuracy and performance.

The paper also emphasizes the crucial function that tokenization serves in NLP, paving the way for developments in fields like text classification, named entity identification, and sentiment analysis. As a result, additional research in tokenization has the potential to completely change the NLP field by allowing the creation of more precise and efficient models for a variety of applications.

This paper provides a comprehensive review of tokenization, identifies key challenges and restrictions, and highlights new research opportunities, making it a very useful resource for NLP researchers.

Summaryofkeyfindings

The key findings of this comprehensive review of NLPtokenizers include:

1. Tokenization is a crucial NLP procedure that divides the text into useful chunks to facilitate subsequent processing.
2. Over time, tokenization techniques have significantly advanced, moving from rule-based to statistical and neural network-based models.
3. Word, subword, and character-level tokenization, for example,each has its benefits and restrictions.
4. To choose the best tool for their particular use case, researchers must compare the performance of popular tokenization tools and libraries.
5. Contextual data,unsupervised techniques,and transfer learning are recent developments in tokenization research.
6. Noisy and informal writing, trade-offs between speed and accuracy, and a lack of standardization are all difficulties with tokenization.
7. The field of tokenization research hasa bright future, and advances in machine learningmay pave the way for more advanced tokenization techniques.
8. For NLP tasks like text classification, named entity recognition, and sentiment analysis, tokenization has broader consequences.

This review provides insights into the world of tokenization research and its importance in enabling further advances in NLP.

Contributions of the paper

This study significantly advances the field of NLP tokenization research in many ways. First, a thorough discussion of several tokenization techniques, including rule-based, statistical, and neural network-based methods, is given in the study. Additionally, this paper analyses the benefits and drawbacks of each of these approaches and provides illustrations of well-known tokenization tools and libraries.

Second, this review emphasizes current developments in tokenization research, such as the application of unsupervised techniques for handling noisy and informal writing and the utilization of contextual information. The report also points out many significant issues and constraints in tokenization research, such as the need for more reliable approaches to deal with illegible and noisy text and the trade-offs between accuracy and speed.

Finally, this study offers insightful perspectives on the future of tokenization research by highlighting interesting lines of inquiry and underlining tokenization's broader ramifications for the NLP communityas a whole. As a result, this work is a useful tool for NLP researchers, providing a thorough reference for the most recent advancements and trends in tokenization research.

Limitations and future work

Although the state-of-the-art tokenization methods are thoroughly reviewed in this study, there are still some drawbacks and potential research areas.

The effectiveness of current tokenization approaches in noisy and informal language is one of their key drawbacks. Although unsupervised and contextual-based approaches have made considerable strides in tackling this problem, more study is required to increase the accuracy of these techniques.

As many of the existing techniques and tools are mostly targeted at English text, there is also a need to investigate tokenization for various languages and language kinds. Future study in this field has great potential because tokenization is a critical NLP step for handling text in various languages.

In real-time applications where speed is essential, it is necessary to investigate the trade-offs between speed and accuracy in tokenization approaches. The most effective and reliable tokenization techniques for various types of text data require further study.

References

1. Sebastiani,F.(2002).Machinelearninginautomatedtextcategorization. *ACMcomputing surveys (CSUR)*, *34*(1), 1-47.
2. Sebastiani,F.(2002).Machinelearninginautomatedtextcategorization. *ACMcomputing surveys (CSUR)*, *34*(1), 1-47.
3. Cavnar,W. B., &Trenkle,J.M.(1994, April).N-gram-basedtextcategorization. In *Proceedingsof SDAIR-94, 3rd annual symposium on document analysis and information retrieval* (Vol. 161175).
4. Goddard,C.(2011). *Semanticanalysis:Apracticalintroduction*.OxfordUniversity Press.
5. Dumais,S.T.(2004).Latentsemantic analysis.*Annu.Rev.Inf.Sci.Technol.*,*38*(1),188-230.
6. Landauer,T.K.,Foltz,P. W.,&Laham,D.(1998).Anintroductiontolatentsemantic analysis. *Discourse processes*, *25*(2-3), 259-284.
7. Koehn,P.(2009).*Statisticalmachinetranslation*.CambridgeUniversityPress.
8. Lopez,A.(2008).Statisticalmachinetranslation.*ACMComputingSurveys(CSUR)*,*40*(3),1-49.
9. Sarawagi,S.(2008).Informationextraction.*FoundationsandTrends®inDatabases*,*1*(3),261-377.
10. Cowie,J., &Lehnert,W.(1996).Information extraction. *CommunicationsoftheACM*,*39*(1),80- 91.
11. Vaswani,A., Shazeer,N.,Parmar, N.,Uszkoreit,J.,Jones,L.,Gomez,A.N.,... &Polosukhin,I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.
12. Grefenstette,G.(1999).Tokenization.*SyntacticWordclassTagging*,117-133.
13. Marcus,M.P.,Santorini,B.,&Marcinkiewicz,M.A.(1993).Buildingalargeannotatedcorpusof English. The Penn Treebank. Computational Linguistics, 19, 313 – 330
14. StevenBirdandEdwardLoper.NLTK: TheNaturalLanguageToolkit.InProceedingsofACL2004 on Interactive poster and demonstration sessions, page 31. ACL, 2004.
15. Manning,C. D., Surdeanu, M.,Bauer,J.,Finkel,J.R., Bethard, S., &McClosky,D.(2014,June). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations* (pp. 55-60).
16. Berger, A., DellaPietra, S. A., &DellaPietra,V.J.(1996).A maximumentropy approachto natural language processing. *Computational linguistics*, *22*(1), 39-71.
17. Wallach,H.M.(2004).Conditionalrandomfields:Anintroduction.*TechnicalReports(CIS)*,22.
18. Altinok,D.(2021). *MasteringspaCy:Anend-to-endpracticalguidetoimplementingNLP applications using the Python ecosystem*. Packt Publishing Ltd.

19. Jain,H.J.,Bewoor,M.S., &Patil,S.H.(2012).Contextsensitive textsummarization usingk means clustering algorithm. *Int. J. Soft Comput. Eng*, *2*(2), 301-304.
20. Białecki,A.,Muir,R.,Ingersoll,G.,&Imagination,L.(2012,August).Apachelucene4.In *SIGIR 2012 workshop on open source information retrieval* (p. 17).
21. Varghese, R., & Jayasree, M. (2013). A survey on sentiment analysis and opinion mining.*InternationaljournalofResearchinengineeringandtechnology*,*2*(11),312-317.
22. Bikel,D., &Zitouni,I.(2012). *Multilingualnaturallanguageprocessingapplications:fromtheoryto practice*. IBM Press.
23. Dunning,T.(1994). *Statisticalidentificationoflanguage*(pp.940-273).LasCruces:Computing Research Laboratory, New Mexico State University.
24. Zhu, X., Zhu, J., Li, H., Wu, X., Li, H., Wang, X., & Dai, J. (2022). Uni-perceiver: Pre-training unifiedarchitectureforgenericperceptionforzero-shotandfew-shottasks.In *Proceedingsofthe IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 16804-16815).
25. Tayefi,M.,Ngo,P.,Chomutare,T.,Dalianis,H.,Salvi,E.,Budrionis,A.,&Godtliebsen,F.(2021). Challengesandopportunitiesbeyondstructureddatainanalysisofelectronichealth records. *Wiley Interdisciplinary Reviews: Computational Statistics*, *13*(6), e1549.
26. Oda, Y., Fudaba, H., Neubig, G., Hata, H., Sakti, S., Toda, T., & Nakamura, S. (2015, November).Learningtogeneratepseudo-codefromsourcecodeusingstatisticalmachine translation. In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)* (pp. 574-584). IEEE.
27. Savova,G. K.,Masanz,J. J.,Ogren, P.V.,Zheng, J.,Sohn, S.,Kipper-Schuler, K.C., &Chute, C. G. (2010). Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture,componentevaluationandapplications.*JournaloftheAmericanMedicalInformatics Association*, *17*(5), 507-513.
28. Flach,P.,&Kull,M.(2015).Precision-recall-gaincurves:PRanalysisdoneright. *Advancesin neural information processing systems*, *28*.
29. Alyafeai,Z.,Al-shaibani,M.S.,Ghaleb,M.,&Ahmad,I.(2022).Evaluatingvarious tokenizersfor Arabic text classification. *Neural Processing Letters*, 1-23.
30. Kaur, H.,&Maini,R.(2019).Assessing lexicalsimilarity betweenshortsentences ofsourcecode based on granularity. *International Journal of Information Technology*, *11*, 599-614.
31. Carreras,X., &Màrquez,L. (2005, June).IntroductiontotheCoNLL-2005sharedtask:Semantic role labeling. In *Proceedingsoftheninthconferenceoncomputational natural languagelearning (CoNLL-2005)* (pp. 152-164).
32. Sang, E.F., &Buchholz, S.(2000). Introductiontothe CoNLL-2000sharedtask: Chunking. *arXiv preprint cs/0009008*.
33. Hajic,J.,Ciaramita, M.,Johansson,R.,Kawahara,D., Martí, M.A.,Màrquez, L.,...&Zhang,Y. (2009, June). The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task* (pp. 1-18).
34. Nivre,J.,Hall,J., Kübler, S.,McDonald,R., Nilsson,J., Riedel,S.,& Yuret,D. (2007,June).The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)* (pp. 915-932).
35. Laali,M.,Cianflone,A.,&Kosseim,L.(2017).Theclac discourseparseratconll-2016. *arXiv preprint arXiv:1708.05798*.
36. Radhakrishnan,P.,Talukdar,P.,&Varma,V.(2018,June).Elden:Improvedentitylinkingusing densified knowledge graphs. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (pp. 1844-1853).
37. Onoe,Y.,&Durrett,G.(2020,April).Fine-grainedentitytypingfordomainindependententity linking. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 34, No. 05, pp. 8576-8583).
38. Kudo,T.,&Richardson,J.(2018).Sentencepiece:Asimpleandlanguageindependentsubword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

39. Narayan, R.,Chakraverty, S.,&Singh, V. P.(2014).Neural networkbasedparts ofspeech tagger for hindi. *IFAC Proceedings Volumes*, *47*(1), 519-524.

40. Suykens,J. A., &Vandewalle,J.(1999).Leastsquaressupportvectormachine classifiers. *Neural processing letters*, *9*, 293-300.

41. Zhai, C., & Lafferty, J. (2017, August). A study of smoothing methods for language models appliedtoadhocinformationretrieval.In *ACMSIGIR Forum*(Vol.51,No.2,pp.268-276).New York, NY, USA: ACM.

42. Sutskever,I., Martens,J., &Hinton,G. E.(2011).Generatingtextwithrecurrentneuralnetworks. In *Proceedings of the 28th international conference on machine learning (ICML-11)* (pp. 1017- 1024).

43. Devlin,J.,Chang,M.W.,Lee, K., &Toutanova, K.(2018).Bert: Pre-trainingof deepbidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

44. Řehůřek, R., & Sojka, P. (2011). Gensim—statistical semantics in python. *Retrieved from genism. org.*

45. Abadi, M., Barham, P., Chen,J.,Chen,Z.,Davis, A., Dean,J., ... &Zheng, X.(2016,November). Tensorflow: a system for large-scale machine learning. In *Osdi* (Vol. 16, No. 2016, pp. 265-283).

46. Wolf,T., Debut,L.,Sanh,V.,Chaumond,J., Delangue, C.,Moi,A., ... &Rush,A. M.(2019). Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

47. Ketkar,N.,& Ketkar,N.(2017).Introduction tokeras. *Deeplearningwithpython:ahands-on introduction*, 97-111.

48. Vedaldi,A.,&Lenc,K.(2015,October).Matconvnet: Convolutionalneuralnetworksformatlab. In *Proceedings of the 23rd ACM international conference on Multimedia* (pp. 689-692).

49. Petrov,Slav,DipanjanDas,andRyanMcDonald."Auniversalpart-of-speechtagset."*arXiv preprint arXiv:1104.2086* (2011).

50. Ng, H. T., & Low, J. K. (2004, July). Chinese part-of-speech tagging: One-at-a-time or all-at-once?word-basedorcharacter-based?. In *Proceedingsofthe2004ConferenceonEmpirical Methods in Natural Language Processing* (pp. 277-284).