Hybrid Ensemble-Rule Algorithm for Improved MEDLINE® Sentence Boundary Detection

Daniel X. Le, James G. Mork, and Sameer Antani
Lister Hill National Center for Biomedical Communications
National Library of Medicine
8600 Rockville Pike, Bethesda, MD 20894

Abstract

Sentence boundary detection (SBD) is a fundamental building block in the Natural Language Processing (NLP) pipeline. Incorrect SBD may impact subsequent processing stages resulting in decreased performance. In well-behaved corpora, a few simple rules based on punctuation and capitalization are sufficient for successfully detecting sentence boundaries. However, a corpus like MEDLINE citations presents challenges for SBD due to several syntactic ambiguities, e.g., abbreviation-periods, capital letters in first words of sentences, etc. In this manuscript we present an algorithm to address these challenges based on majority voting among three SBD engines (Python NLTK, pySBD, and Syntok) followed by custom post-processing algorithms that rely on NLP spaCy part-of-speech, abbreviation and capital letter detection, and computing general sentence statistics. Experiments on several thousand MEDLINE citations show that our proposed approach for combining multiple SBD engines and post-processing rules performs better than each individual engine.

Introduction

This paper describes an ongoing effort at the National Library of Medicine (NLM) related to the detection of sentence boundaries of MEDLINE documents where a document refers to a MEDLINE citation which contains a title and may contain an abstract.

Sentence boundary detection (SBD) is an important pre-processing step in many Natural Language Processing (NLP) applications, such as sentence embedding, part-of-speech taggers, document indexing, and question answering. Several techniques for detecting sentence boundaries have been reported in the NLP literature covering a variety of text domains including general text (WSJ text and the Brown corpus [1,2]), biomedical text (the GENIA corpus [3] of biomedical abstracts), clinical text (the i2b2 corpus [4]), and legal text (the United States Courts decisions dataset [5]).

As pointed out in reviews by Griffis [6] and Read [7], the performance of the reviewed SBD techniques is domain dependent. Most do well in general text domains that "conform to formal English" but perform poorly in certain domains, like biomedicine and legal text, where customized algorithms tend to perform better. A specialized corpus like MEDLINE containing journal citations, titles and abstracts presents challenges for SBD due to the difficulties in recognizing uppercase letters belonging to domain-specific (genetic, biological, and chemical) terms, and disambiguating abbreviation-periods among journal abbreviations and medical terms.

In the domain of general text SBD, Riley [8] used a decision tree classifier while Gillick [9] used a supervised approach with Naïve Bayes and Support Vector Machine (SVM). Kiss [10] proposed an unsupervised abbreviation detection algorithm with the Punkt model that "can detect abbreviations based on collocation between periods and truncated words". Palmer [11] used a neural network with inputs as vectors of binary values representing contextual words surrounding punctuation marks. These techniques reported high accuracies of over 98.00% on both WSJ and the Brown corpus.

Savelka [5] and Sanchez [12] suggested methods to detect sentence boundaries in legal text domains "using customized Punkt tokenizer and Conditional Random Field (CRF) algorithms". Both systems reported accuracies in the range from 75.00% to 90.00%. Buyko [13] recommended a new approach by retraining the OpenNLP software application on specific biology domains, including the GENIA and PennBioIE25 corpora, and the system reported accuracies of 99.00% and 97.40%, respectively. This experiment was applied on the GENIA corpus [3], a collection with 1,999 abstracts from the MEDLINE database. Kreuzthaler [14] suggested a system using two SVM binary classifiers for abbreviation and sentence detection to detect sentence boundaries in German clinical narratives. The system reported an F-measure of 95.00% for abbreviation detection and 94.00% for sentence delineation.

In this paper, we describe our hybrid-ensemble approach to detect sentence boundaries to address and resolve identified SBD challenges. The algorithm is based on majority voting among three SBD engines (Python NLTK [15], pySBD [16], and Syntok [17]) followed by custom post-processing rule-based algorithms to detect sentence boundaries that rely on NLP spaCy part-of-speech (POS) [18] tokenization library, abbreviation, and capital letter detection, and computing general sentence statistics. The performance of the proposed SBD algorithm has been evaluated using two data sets of approximately three thousand MEDLINE citations. The evaluation results show that our hybrid-ensemble approach performs better than each individual engine in terms of improving the detection accuracy of correct sentence boundaries.

Methods

Data sets

The training data set consists of 1,514,446 citations randomly selected from the 30-million citations in the 2020 MEDLINE Baseline data set [19] and is used for observations and for developing features and rules for the proposed SBD algorithm. There are two ground-truth data sets that we used to evaluate the system performance of the proposed SBD algorithm: the GENIA corpus [3] ground-truth data set and the 2021 SBD ground-truth data set.

The GENIA corpus [3] ground-truth data set consists of 18,541 sentences segmented from 1,999 citations from MEDLINE database. It was created from the Genome Informatics Extraction (GENIA) project to provide the gold standard for the evaluation of text mining systems. For the 2021 SBD ground-truth data set, we downloaded approximately 32-million citations from the 2021 MEDLINE Baseline data set [19] in XML file format. We then randomly selected 1,020 citations that were not in the 2020 MEDLINE Baseline data set and extracted titles and abstracts from the XML. For any "structured abstracts" citations, we kept only the label followed by a colon space and filtered out "NlmCategory" and their XML tag "AbstractText" [20]. For example, the XML text "<AbstractText Label="METHODS" NlmCategory="UNASSIGNED">This is a test.</AbstractText> became "METHODS: This is a test." From the selected 1,020 citations, we manually annotated the boundaries of 11,204 sentences to build the 2021 SBD ground-truth data set.

Basic definitions

Definitions of the basic features used in our system are given here.

1. A "regular" capital word versus a "special" capital word: In our system, words that begin a sentence could be considered as "regular" or "special" capital words. A "regular" capital word has its first character as an uppercase letter (e.g. "National") while a "special" capital word includes an uppercase letter after its first character (e.g. "20-Hydroxycholesterol").

2. VERB-Phrase sentences and NOUN-Phrase sentences: POS tagging is a technique to map words as nouns, verbs, adjectives, adverbs, etc. In our system, NLP spaCy POS tagging library [18] is used to generate two lists of attributes of the tokenization outputs for any sentence: a POS list and a TAG list. A POS list shows the coarse-grained part of speech, and its associated TAG list shows the fine-grained part of speech, as shown in the following example.

Sentence:	"This	method	shows	advantages	in	two	aspects."	
POS list:	['DET',	'NOUN',	'VERB',	'NOUN',	'ADP',	'NUM',	'NOUN',	'PUNCT']
TAG list:	['DT',	'NN',	'VBZ',	'NNS',	'IN',	'CD',	'NNS',	'.']

A complete list of POS and associated TAG list are available [18]. Based on these lists, a "VERB-Phrase sentence" and a "NOUN-Phrase sentence" are defined as follows:

A sentence is considered a "VERB-Phrase sentence" if it satisfies one of the following two cases:
Case 1: POS = AUX and TAG = VB or VBD or VBP or VBZ
Case 2: POS = VERB and TAG = VBD or VBP or VBZ or VBN or MD

A sentence is considered a "NOUN-Phrase sentence" if it satisfies one of the following three cases:
Case 1: POS = NOUN or PROPN
Case 2: POS = PRON and TAG = EX or PRP or WP
Case 3: POS = DET and TAG = WP$

As a result, we define the "**five sentence-based POS tagging types**" as follows:

NP:	A sentence with only a "NOUN-Phrase sentence", *e.g., "Level of evidence : Level II."*
VP:	A sentence with only a "VERB-Phrase sentence", *e.g., "were investigated."*
NP+VP:	A sentence with a "NOUN-Phrase sentence" followed by a "VERB-Phrase sentence", *e.g., "Fast and scalable tools are hence needed"*
VP+NP:	A sentence with a "VERB-Phrase sentence" followed by a "NOUN-Phrase sentence", *e.g., "Excluded were 139 Web sites."*
NP+VP+NP:	A sentence with a "NOUN-Phrase sentence" followed by a "VERB-Phrase" and a "NOUN-Phrase" *e.g., "Costs were estimated in US dollars."*

3. Normal POS versus Special POS: Based on our observations, we recognized two POS terms that play an important role in helping to make the connection decisions between consecutive sentences: “PROPN” and “INTJ”. They are defined in this paper as **special-POS** and the remaining terms are **normal-POS** (“ADJ”, “ADP”, “ADV”, “AUX”, “CCONJ”, “DET”, “NOUN”, “NUM”, “PART”, “PRON”, “PUNCT”, “SCONJ”, “SYM”, “VERB”, and “X”). The following show examples of broken sentences that could be connected through the help of **special-POS**.

By contrast, cyto. ADP, NOUN, PUNCT, PROPN, PUNCT	o, i.e., the ... tension. INTJ, PUNCT, X, PUNCT, DET, ..., NOUN, PUNCT
Furthermore, ... anion (O2-.) ADV, PUNCT, ..., NOUN, PUNCT, INTJ, PUNCT, PUNCT	with 1 ... scavenging O2-. ADP, NUM, ..., VERB, X, PUNCT
Orv Hetil. PROPN, PROPN, PUNCT	2020; 161(1): 33-38. NUM, PUNCT, NUM, PUNCT, PUNCT, NUM, SYM, NUM, PUNCT

Algorithms to Handle Sentence Boundaries Challenges in MEDLINE

We used two algorithms to address the above-mentioned challenges: the “special” capital word detection algorithm and the abbreviations detection algorithm.

1. The “special” capital word detection algorithm: A sentence usually starts with a word in which its first character is an uppercase letter. However, this condition might not hold true for some genetic, biological, and chemical terms where digits, punctuation, lowercase letters, and symbols (such as “12α-Hydroxyl”, “(-)-Deprenyl”, “•Lichen”) might start a sentence. For these terms, we observed some special patterns from characters before their first uppercase letter (such as “digits, Greek letters, punctuations” for “12α-Hydroxyl”, “punctuations” for “(-)-Deprenyl”, “symbols” for “•Lichen”). Therefore, in order to address such variants, we built a predefined list of “special” capital word patterns and developed an algorithm using this list to detect whether the first word of a sentence is a “special” capital word.

To build a predefined list of capital word patterns, we collected only sentences of which the first word has at least one uppercase letter, but its first character is not an uppercase letter. In addition, we limited the sentence length to be greater or equal to the average sentence length which is 110 characters to reduce the number of collected samples.

> *Note that a common plain English guideline says that an average sentence length is about 15–20 words [21] and the average word length is about 4.7 characters [22]. In this paper, the average sentence length (AVGSL) is calculated based on 5 characters per word, an average of 18 words, and 17 spaces per sentence as follows: AVGSL = 18 words * 5 characters + 17 spaces ≈ 110 characters.*

To build a predefined list of capital word patterns, we first collected all “special” capital word patterns that are generated by (a) taking the sentences’ first words and truncate them at their uppercase letter. For example, the first word “12α-Hydroxyl” is truncated into “12α-”, (b) building lists of Unicode names associated with characters of the truncated words. For the truncated word “12α-”, the list of Unicode names is: “DIGIT ONE”, “DIGIT TWO”, “GREEK SMALL LETTER ALPHA”, “HYPHEN-MINUS”, (c) replacing Unicode names based on their type by their predefined “replacement” symbols using the replacement table (Table 1). For the list of Unicode names in b), the replacement symbols are **00XH**, (d) replacing consecutive similar “replacement” symbols by a single “replacement” symbol to build “special” capital word patterns for the truncated words. For the above replacement symbols in c), the “special” capital word pattern is **0XH**. We then tabulated and sorted the collection of the “special” capital word patterns based on their pattern occurrence frequency and selected the topmost patterns as the predefined list of “special” capital word patterns.

Table 1. Replacement symbols

Type of Unicode Name	Replacement	Type of Unicode Name	Replacement
Greek alphabet	**X**	Small letter	**Y**
Hyphen	**H**	Bullet	**B**
Punctuation	**P**	Inverted question	**I**
Digit or Number	**0**	Symbols	**Z**
Superscript	**S**	Special space	*****
Roman numeral	**R**	No replacement	**?**

We applied the above procedure on the training data set described in the Methods section and came up with a predefined list of “special” capital word patterns (Table 2).

A first word in a sentence could be classified as a “special” capital word or not by first represent it as a symbol-based word using the replacement table (Table 1) and then compare the symbol-based word with the predefined list of "special" capital word patterns (Table 2).

The algorithm to detect a "special" capital word for the first word in a sentence starts by truncating the given word at its first uppercase letter and then represents the truncated word as a list of Unicode names where each Unicode name is associated with each character of the truncated word. Next, it converts a list of Unicode names into a symbol-based word using the replacement table (Table 1). Finally, it checks the symbol-based word against the predefined list of "special" capital word patterns (Table 2) to determine whether or not the given word is a "special" capital word.

Table 2. Predefined list of "special" capital word patterns

Sentence First Words	**"Special" Capital Word Patterns**
Start with digits	"0H", "0", "0P0H", "0PH", "0HP0H", "0HP", "0P0PH", "0P0P0H", "0XH", "0P0P0P0H", "0HP0P0H", "0HXH", "0HP0HP0H", "0P0P0P0PH", "0HP0"
Start with punctuations	"P", "H", "PHPH", "PH", "PHP"
Start with lowercase letter	"XH", "X0H", "XP0PH"
Start with symbols or others	"Z", "B", "S", "*", "I", "ZH", "RP"

2. The abbreviations detection algorithm: Based on our observations of MEDLINE citations in our training data set, we have found that one of the ways sentences are incorrectly segmented is due to the failure of recognizing abbreviations between consecutive sentences. To minimize the impact of this problem, we build a predefined list of abbreviations (Figures 1 and 2) from last words of sentences and develop a simple abbreviations detection algorithm using it.

Abbreviations could connect to sentences that have the first character of their first word as either a lowercase or uppercase letter; however, we limit the algorithm's attention to just lowercase letters and consider only abbreviations having 8 characters or less to reduce the number of collected samples.

To build a list of abbreviations, we collected all words that (a) have 8 characters or less, (b) are ended with a period, and (c) are followed with a word starting with a lowercase letter. We then tabulated the collection of the collected words into two groups: 1-period group (e.g., "approx.", "Corp.", etc.) and 2-period group (e.g., "i.e.", "Ph.D.", etc.) and filtered out words with a frequency of less than 100 to build the predefined list of abbreviations.

We applied the above procedure on the training data set and came up with the following list of abbreviations for the 1-period group (Figure 1) and the 2-period group (Figure 2).

'(approx.', '(ca.', '(exp.', '(fig.', '(max.', '(mol.', '(ref.', '(Tab.', '(vs.', 'approx.', 'eg.', 'eq.', 'equiv.', 'etc.', 'fig.', 'figs.', 'gm.', 'inc.', 'incl.', 'lb.', 'nmol.', 'ref.', 'resp.', 'subsp.', 'vol.', 'vs.', 'et al.', '(no.', '(pp.', 'appr.', 'no.', 'nos.', 'pg.', 'pp.', 'Ae.', 'Aer.', 'aff.', 'An.', 'ca.', 'cc.', 'cf.', 'cv.', 'Cx.', 'Exp.', 'int.', 'Jr.', 'microg.', 'microM.', 'O2-.', 'Pr.', 'Ps.', 'pv.', 'Rh.', 'Rps.', 'Rs.', 'sc.', 'Sch.', 'Ser.', 'Sh.', 'sp.', 'spp.', 'sq.', 'ss.', 'ssp.', 'St.', 'Staph.', 'str.', 'Strep.', 'Th.', 'Tr.', 'Vmax.', 'Chem.', 'Biol.', 'Biochem.', 'Virol.', 'Biophys.', 'anti-A.' to 'anti-Z.'

Figure 1: The "general" 1-period abbreviation group

'(e.g.', '(i.e.', 'b.wt.', 'e.g.', 'f.sp.', 'i.e.', 'mg./kg.', 'mol.wt.', 'mol.wts.', 'ng./ml.', 'p.m.', 'Ph.D.', 'U.K.', 'U.S.', '(i.p.', '(n.s.', '(p.i.', '(s.d.', 'a.m.', 'b.d.', 'b.w.', 'c.i.', 'i.d.', 'i.g.', 'i.m.', 'i.p.', 'i.pl.', 'i.t.', 'i.th.', 'i.v.', 'm.w.', 'n.s.', 'n.sp.', 'p.c.', 'p.i.', 'p.o.', 's.c.', 's.d.', 's.e.', 's.l.', 's.s.', 'sp.n.'

Figure 2: The "general" 2-period abbreviation group

The simple abbreviations detection algorithm could be implemented by checking the sentence last word. If the sentence last word without trailing whitespace characters is found in the above 1-period or 2-period abbreviation group, then it is an abbreviation.

System Overview

The algorithm starts with segmenting a document into sentences with each SBD engine, followed by a majority voting to build a list of most voted sentence "proposals" for the document. After this, the content-based features and the grammar-based features are computed using the five sentence-based POS tagging types, the abbreviation detection algorithm, the capital word detection algorithm, and computing general sentence statistics. These features are then used to determine whether to connect consecutive sentences. The algorithm ends with splitting sentences with specific patterns.

To decide on the connection among segmented sentences, the algorithmic approach is to look for complete and incomplete sentences. Incomplete sentences could be the results of SBD engines that failed to recognize sentence delimiters, identify capital words, detect abbreviations, etc. Further, incomplete sentences should connect to other sentences.

We used sentence attributes, the number of votes on each sentence, sentence structure, and POS to evaluate the completeness of each segmented sentence. There are two sets of features for each sentence: content-based and grammar-based. Content-based features are based on character statistics, abbreviations, majority voting, and five sentence-based POS tagging types. Grammar-based features are based on the POS tags for the first and last two terms in a sentence. These two sets of features complement each other to help in making connection decisions between consecutive segmented sentences. Figure 3 shows the workflow of the MEDLINE SBD system to process a MEDLINE citation and each step of this process is described in detail in the next section.

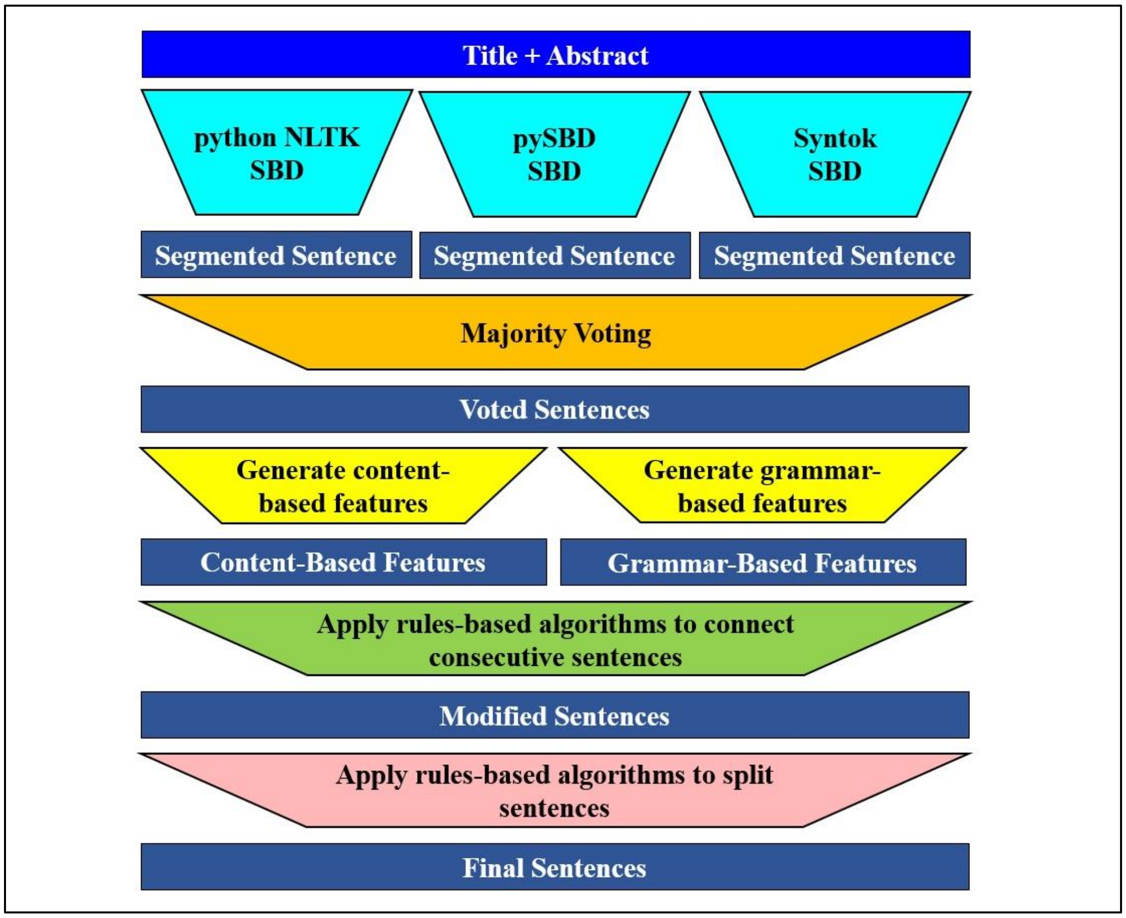


Figure 3: MEDLINE Sentence Boundary Detection Diagram

MEDLINE Sentence Boundary Detection Process

The MEDLINE SBD process consists of six steps: (1) segment a document into sentences by each SBD engine, (2) build voted sentences among SBD engines using a majority voting algorithm, (3) generate the content-based features for voted sentences, (4) generate the grammar-based features for voted sentences, (5) apply the rules-based SBD algorithm to decide on connection between consecutive sentences, and (6) split sentences with specific patterns.

1. Segment a document into sentences: Each SBD engine segments the document into sentences and each sentence is recorded with a starting index and a stopping index relative to the start of the document and its voting SBD engine.

2. Build voted sentences using a majority voting algorithm: The majority voting algorithm groups sentences from SBD engines together and assigns the number of votes for each sentence to build the voted sentences for the document. For the three SBD engines, the algorithm collects sentences into three groups: 3-vote group, 2-vote group, and 1-vote group. First, sentences with 3 votes are collected into the 3-vote group. Next, sentences with 2 votes that are not in the 3-vote group are collected into the 2-vote group. Finally, sentences with 1 vote that are not in the 3-vote and 2-vote groups are collected into the 1-vote group.

Note that only one SBD engine will be selected for the last group with one vote to avoid overlapping among selected segmented sentences. Based on the observations of segmented sentences by the three SBD engines over several thousand MEDLINE citations, the priority for which engine to select is Python NLTK, pySBD, and Syntok.

3. Generate the sentence content-based features: The content-based features consist of structure information and character statistics of a sentence, and they are used to justify if a sentence is complete or incomplete for the purpose of connecting consecutive sentences. There are six sentence content-based features: "**GRAMMAR**", "**STYLE**", "**ABBREVIATION**", "**LENGTH**", "**LOCATION**", and "**CATEGORY**", and they are defined and explained next.

A sentence is complete when it begins with a capital letter (a regular/special capital word), ends with a punctuation mark (period, question mark, or exclamation point), and has both a subject and a verb. A subject may be a noun (a person, place, or thing) or a pronoun. For example, a sentence that starts with a capital word and consists of one or several combinations of "NOUN-Phrase sentence" and "VERB-Phrase sentence" has a high chance to be a complete sentence. As a result, we used the five sentence-based POS tagging types and the "special" capital word detection algorithm defined in the Methods section (*Basic definitions and Algorithms to Handle Sentence Boundaries Challenges in MEDLINE*) to generate the "**GRAMMAR**" feature and the "**STYLE**" feature, respectively.

GRAMMAR = "Grammatical Complete"	If the POS tagging type = **NP+VP+NP** or **NP+VP** or **VP+NP**
GRAMMAR = "Grammatical Incomplete"	If the POS tagging type = **NP** or **VP** or **Unknown**

STYLE = "Uppercase Sentence"	If a sentence has a "regular" or "special" capital word
STYLE = "Lowercase Sentence"	Otherwise

Next, we apply the abbreviations detection algorithm defined in the Methods section (*Algorithms to Handle Sentence Boundaries Challenges in MEDLINE*) on the last word of the sentence to build the "**ABBREVIATION**" feature.

ABBREVIATION = "Yes"	If an abbreviation is found at the end of the sentence
ABBREVIATION = "No"	Otherwise

Based on our observations, a sentence with an average sentence length (AVGSL $\approx$ 110 characters) or higher is most likely to be a complete sentence. In addition, the higher number of votes assigned on a sentence from three SBD engines gives more confidence that it is a complete sentence. Therefore, the number of characters and the number of votes are used to define a sentence content-based feature called "**LENGTH**" as follows:

LENGTH = "Medium-or-Long Sentence"	If (votes <= 2 and number of characters >= ¾ of AVGSL) or (votes > 2 and number of characters >= ½ of AVGSL)
LENGTH = "Short Sentence"	Otherwise

The location of a sentence is another feature, and its values include below:

LOCATION = "First"	If the sentence is the first sentence
LOCATION = "Last"	If the sentence is the last sentence
LOCATION = "Middle"	Otherwise

In addition, the punctuations are used to define one more feature: **"CATEGORY".**

CATEGORY = "Punctuations-Only-Sentence"	If the sentence consists of punctuations only
CATEGORY = "Regular"	Otherwise

4. Generate the sentence grammar-based features: POS categorizes words by type, such as nouns, verbs, adjectives, and each sentence is associated with a POS list that is used to build the grammar-based features. As shown in the Methods section (*Basic definitions*), some combinations between the ending POS terms of the current sentence and the beginning POS terms of the next sentence could help to connect them together. Therefore, we define two grammar-based features for each sentence: "**BEGINNING POST TERMS (BPT)**" and **"ENDING POST TERMS (EPT)**". The first feature is based on the first one or two POS terms of the sentence while the second one is based on the last one or two POS terms of the sentence.

The four values of the **BPT** are "normal-POS", "PUNCT+normal-POS", "special-POS", and "PUNCT+special-POS".

normal-POS	The POS list starts with a normal-POS
PUNCT+normal-POS	The POS list starts with a PUNCT followed with a normal-POS
special-POS	The POS list starts with a special-POS ("PROPN" or "INTJ")
PUNCT+special-POS	The POS list starts with a PUNCT followed with a special-POS

The four values of the **EPT** are “normal-POS”, “normal-POS+PUNCT”, “special-POS”, and “special-POS+PUNCT”.

normal-POS	The POS list ends with a normal-POS
normal-POS+PUNCT	The POS list ends with a normal-POS followed with a PUNCT
special-POS	The POS list ends with a special-POS
special-POS+PUNCT	The POS list ends with a special-POS followed with a PUNCT

For example, consider the following sentence with its POS list:

Sentence: “This method shows advantages in two aspects.”
POS list: ['DET', 'NOUN', 'VERB', 'NOUN', 'ADP', 'NUM', 'NOUN', 'PUNCT']

The **BPT** feature is “**normal-POS**” because its POS list starts with a **normal-POS** ('DET') and the **EPT** feature is “**normal-POS+PUNCT**” because its POS list ends with a **normal-POS** and a **PUNCT** ('NOUN', 'PUNCT').

5. Apply the rules-based SBD algorithm: We apply the rules-based SBD algorithm to connect or separate segmented sentences and the following statements are used in the algorithm.

- Set the sentence “START” or “STOP” means it is the start sentence or the end sentence, respectively.
- Set the sentence “RIGHT CONNECT” means it connects to the next sentence on its right.
- Set the sentence “LEFT CONNECT” means it connects to the previous sentence on its left.

There are heuristic rules and three parts of observation rules, and they are executed in this order. In addition, there are synchronization rules to synchronize connections.

5a) Heuristic Rules: There are eight straightforward heuristic rules, and they are created to handle common-sense cases. For example, sentences ended with abbreviations should be “**RIGHT CONNECT**” to the next sentence.

```
Rule 1: If sentence CATEGORY = “Punctuations-Only-Sentence” then
            If sentence LOCATION = “First” then set the sentence to “RIGHT CONNECT”
            Else if sentence LOCATION = “Last” then set the sentence to “LEFT CONNECT”
            Else set the sentence to “LEFT CONNECT” and “STOP”
Rule 2: If sentence LOCATION = “First” then set the sentence to “START”
Rule 3: If sentence LOCATION = “Last” then set the sentence to “STOP”
Rule 4: If sentence LENGTH = “Medium-or-Long Sentence” then set the sentence to “START”
Rule 5: If sentence ABBREVIATION = “Yes”
            If sentence LOCATION = “First” or “Middle” then set the sentence to “RIGHT CONNECT”
Rule 6: If the sentence starts with a case-insensitive “Copyright” or a symbol ® then
            If the sentence contains a year between 1900 and 2200 then then set the sentence to “START”
Rule 7: If sentence LENGTH = “Short Sentence” and GRAMMAR = “Grammatical Complete” then
            If sentence STYLE = “Uppercase Sentence” then set the sentence to “START”
Rule 8: If sentence LENGTH = “Short Sentence” and GRAMMAR = “Grammatical Incomplete” and
            If number of words <= 1 or number of words without stop words <= 1 or number of characters <= 5 then
                If sentence LOCATION = “First” then set the sentence to “RIGHT CONNECT”
                Else if sentence LOCATION = “Last” then set the sentence to “LEFT CONNECT”
```

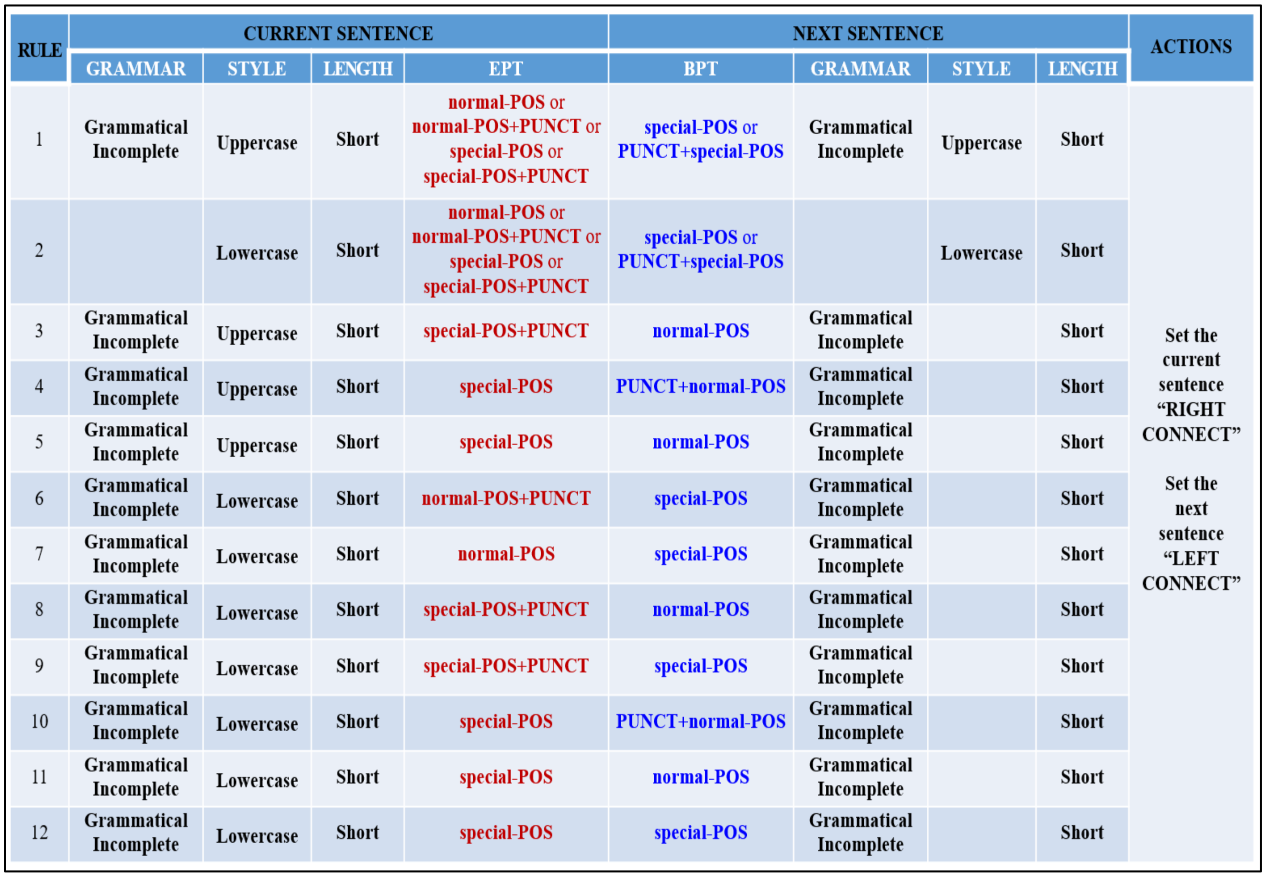
5b) Synchronization rules: These rules are straightforward rules, and they are required to synchronize connections among surrounding sentences when there is a new connection between sentences.

```
Rule 1: If the current sentence is not the first sentence:
            If it is set to “LEFT CONNECT” then set the previous sentence to “RIGHT CONNECT”
            Else if it is set to “START” then set the previous sentence to “STOP”
Rule 2: If the current sentence is not the last sentence:
            If it is set to “RIGHT CONNECT” then set the next sentence to “LEFT CONNECT”
            Else if it is set to “STOP” then set the next sentence to “START”
```

5c) Observation rules - Part 1: When a sentence is a short sentence and does not start with a capital word, it should connect to its preceding sentence as described in the following two rules.

Rule 1: If the current sentence **GRAMMAR** = “Grammatical Incomplete”
and **STYLE** = “Lowercase Sentence” and **LENGTH** = “Short Sentence” then
Set the current to “**LEFT CONNECT**” and the preceding to “**RIGHT CONNECT**”
Rule 2: If the current sentence **GRAMMAR** = “Grammatical Complete”
and **STYLE** = “Lowercase Sentence” and **LENGTH** = “Short Sentence” then
If the preceding sentence **GRAMMAR** = “Grammatical Incomplete” and **LENGTH** = “Short Sentence” then
Set the current to “**LEFT CONNECT**” and the preceding to “**RIGHT CONNECT**”

5d) Observation rules - Part 2: Rules developed in this section are more complicated than the rest since both content-based features and grammar-based features of the current and next sentences are involved to make the connection decisions. There are many cases where sentences are broken into pieces due to errors in recognizing sentence delimiters, capital words, and abbreviations; most of these sentences are incomplete. There are twelve rules shown in Figure 4 that are created based on the combinations between sentence content-based features **GRAMMAR**, **STYLE**, **LENGTH,** and sentence grammar-based features **EPT** and **BPT** and they are applicable only to “Short” sentences.



RULE	CURRENT SENTENCE				NEXT SENTENCE				ACTIONS
	GRAMMAR	STYLE	LENGTH	EPT	BPT	GRAMMAR	STYLE	LENGTH	
1	Grammatical Incomplete	Uppercase	Short	normal-POS or normal-POS+PUNCT or special-POS or special-POS+PUNCT	special-POS or PUNCT+special-POS	Grammatical Incomplete	Uppercase	Short	Set the current sentence “RIGHT CONNECT” Set the next sentence “LEFT CONNECT”
2		Lowercase	Short	normal-POS or normal-POS+PUNCT or special-POS or special-POS+PUNCT	special-POS or PUNCT+special-POS		Lowercase	Short	
3	Grammatical Incomplete	Uppercase	Short	special-POS+PUNCT	normal-POS	Grammatical Incomplete		Short	
4	Grammatical Incomplete	Uppercase	Short	special-POS	PUNCT+normal-POS	Grammatical Incomplete		Short	
5	Grammatical Incomplete	Uppercase	Short	special-POS	normal-POS	Grammatical Incomplete		Short	
6	Grammatical Incomplete	Lowercase	Short	normal-POS+PUNCT	special-POS	Grammatical Incomplete		Short	
7	Grammatical Incomplete	Lowercase	Short	normal-POS	special-POS	Grammatical Incomplete		Short	
8	Grammatical Incomplete	Lowercase	Short	special-POS+PUNCT	normal-POS	Grammatical Incomplete		Short	
9	Grammatical Incomplete	Lowercase	Short	special-POS+PUNCT	special-POS	Grammatical Incomplete		Short	
10	Grammatical Incomplete	Lowercase	Short	special-POS	PUNCT+normal-POS	Grammatical Incomplete		Short	
11	Grammatical Incomplete	Lowercase	Short	special-POS	normal-POS	Grammatical Incomplete		Short	
12	Grammatical Incomplete	Lowercase	Short	special-POS	special-POS	Grammatical Incomplete		Short	

Figure 4: Observation rules - Part 2

5e) Observation rules- Part 3: The following rule is applicable to short and incomplete sentences and the connection decision is based solely on the sentence grammar-based feature **BPT** of the next sentence.

Rule 1: If the current sentence **GRAMMAR** = “Grammatical Incomplete”
and **STYLE** = “Lowercase Sentence” and **LENGTH** =“Short Sentence” then
If the next sentence **GRAMMAR** = “Grammatical Incomplete” and **LENGTH** = “Short Sentence” then
If the next sentence **BPT** = “normal-POS, PUNCT+normal-POS, special-POS, or PUNCT+special-POS”
Set the current to “**RIGHT CONNECT**” and the next to “**LEFT CONNECT**”

6. Split sentences with specific patterns: Through our observations, we identified some specific patterns that SBD engines failed to recognize, and they are as follows:

Pattern #1: It starts with a space, an uppercase letter, a period, a space, and a capital word such as " B. The" or " A. In"
Pattern #2: It starts with a word, a period, and a capital word such as "test.The" or "station.That"

For both patterns, we observed that when sentences in the left and right of the period are grammatically complete (**GRAMMAR** = "Grammatical Complete") and the POS term of the first word followed the period is **'DET'**, **'ADV'**, **'ADJ'**, **'VERB'**, **'PRON'** or **'NUM'**, sentences could be split at the period.

Therefore, the method to split sentences with specific patterns could start with the detection of the pattern #1 or #2 in a sentence and follow by splitting the sentence into the left sentence and the right sentence at the period. When both sentences are grammatically complete and the POS term of the first word of the right sentence belongs to the above-mentioned POS list, the sentence splitting is acceptable.

Discussion

We implemented the SBD algorithm and conducted experiments on MEDLINE citations randomly selected from several different biomedical journals. The algorithm is a combination between a majority voting among three SBD engines and rule-based post-processing algorithms. Some rules are heuristic, while others are derived from observations and from processing the collected training data set. The SBD algorithm is trained from the 2020 MEDLINE training data set and evaluated from two ground-truth data sets and these data sets are described in detail in the Methods section.

For the evaluation, we measure the SBD performance in terms of accuracy using two criteria: sentence-based and citation-based. For the sentence-based, the accuracy is based on the number of sentences detected by an SBD engine that are found in the ground-truth data set (Tables 3A and 3B). For the citation-based, the accuracy is relied on the number of citations having all their sentences found in the ground-truth data set (Tables 4A and 4B).

Table 3A: The GENIA corpus sentence-based evaluation results

Total GENIA corpus sentences: 18,541	Python NLTK	pySBD	Syntok	MEDLINE SBD
Number of sentences found in the GENIA data set	18,036	18,156	17,916	18,453
Accuracy	97.28	97.92	96.63	**99.53**

Table 3B: The GENIA corpus citation-based evaluation results

Total GENIA corpus citations: 1,999	Python NLTK	pySBD	Syntok	MEDLINE SBD
Citations with all sentences found in the GENIA data set	1,761	1,813	1,707	1,945
Accuracy	88.09	90.70	85.39	**97.30**

Table 4A: The 2021 SBD sentence-based evaluation results

Total 2021 SBD sentences: 11,204	Python NLTK	pySBD	Syntok	MEDLINE SBD
Number of sentences found in the 2021 SBD data set	10,899	11,002	10,901	11,149
Accuracy	97.28	98.20	97.30	**99.51**

Table 4B: The 2021 SBD citation-based evaluation results

Total 2021 SBD citations: 1,020	Python NLTK	pySBD	Syntok	MEDLINE SBD
Citations with all sentences found in the 2021 data set	885	934	861	990
Accuracy	86.76	91.57	84.41	**97.06**

The average time in milliseconds to process a citation on a "DELL Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz with 64.0 GB running 64-bit OS Windows 10 Enterprise" are: 1.21 ms, 2.22 ms, 1.36 ms, and 6.27 ms for Python NLTK, pySBD, Syntok, and MEDLINE SBD, respectively.

The results show that the combined MEDLINE SBD engine offers the best sentence boundary detection accuracy on the ground-truth data sets compared to the other three single SBD engines.

Conclusion

We developed the MEDLINE SBD algorithm using majority voting among multiple SBD engines and post-processing rules derived from content-based and grammar-based sentence features. Evaluation results show that our proposed SBD algorithm yielded good performance in terms of accuracy when compared to each single SBD. Therefore, we conclude that the MEDLINE SBD algorithm provides an improved approach to sentence segmentation for MEDLINE citations. However, we identified two shortcomings that should be resolved to further improve the detection results. First, the algorithm starts with a voting from sentences segmented by each SBD engine and follows with rules to decide whether to connect consecutive sentences. Even though there were some brute-force parsing operations in the last step of the MEDLINE SBD process to split segmented sentences with certain specific patterns. These operations actually just help to minimize the impact of the problem by handling obvious cases, but the problem still remained to be solved. Additionally, when all SBD engines do not agree on detected sentence boundaries, a predefined single engine is selected for the task and the algorithm cannot select other engines that might have better sentence boundaries. As a result, we plan to expand the current approach by including sophisticated methods to split segmented sentences as well as techniques to dynamically select an appropriate single engine when none of the engines agree on detected sentence boundaries.

Acknowledgments

This work was supported by the Intramural Research Program of the U.S. National Institutes of Health (NIH), National Library of Medicine (NLM), and Lister Hill National Center for Biomedical Communications (LHNCBC).

References

1. Marcus M, Santorini B, Marcinkiewicz MA. Building a large annotated corpora of English: The Penn Treebank. Comput Linguist. 1993; 19:313–330.
2. Francis WN, Kucera H. Brown corpus manual [internet]. 1979. Available from: http://icame.uib.no/brown/bcm.html
3. Kim JD, Ohta T, Tateisi Y, Tsujii J. GENIA corpus--a semantically annotated corpus for bio-text mining. Bioinformatics. 2003 Jul; 19(Suppl 1):i180–2.
4. Uzuner Ö, South BR, Shen S, DuVall SL. 2010 i2b2/VA challenge on concepts, assertions, and relations in clinical text. J Am Med Inform Assoc. 2012; 18(5):552–6.
5. Savelka J, Walker VR, Grabmair M, Ashley KD. Sentence boundary detection in adjudicatory decisions in the united states. TAL. 2017; 58:21–45.
6. Griffis D, Shivade C, Fosler-Lussier E, Lai AM (2016). A quantitative and qualitative evaluation of sentence boundary detection for the clinical domain. AMIA Summits on Translational Science Proceedings. 2016; 88.
7. Read J, Dridan R, Oepen S, Solberg J. Sentence boundary detection: A long solved problem? Proceedings of COLING. 2012; Posters 985–994.
8. Riley MD. Some applications of tree-based modelling to speech and language. Speech & Natural Language: 1989;15-18.
9. Gillick D. Sentence boundary detection and the problem with the u.s. NAACL Short '09. 2009; 241–244.
10. Kiss T, Strunk J. Unsupervised multilingual sentence boundary detection. Comput. Linguist. 2006; 32(4):485–525.
11. Palmer DD, Hearst MA. Adaptive multilingual sentence boundary disambiguation. Comput Linguist. 1997; 23(2):241–67.
12. Sanchez G. Sentence boundary detection in legal text. Proceedings of the NLLP Workshop. 2019 June; 31–38.
13. Buyko E, Wermter J, Poprat M, Hahn U. Automatically adapting an NLP core engine to the biology domain. Proc ISMB 2006;65–8.
14. Kreuzthaler M, Schulz S. Detection of sentence boundaries and abbreviations in clinical narratives. BioMed Central Ltd. 2015; 15(Suppl 2):S4.
15. Natural Language Toolkit (nltk) [internet]. Available from http://nltk.org
16. Python Sentence Boundary Disambiguation (pysbd) [internet]. Available from https://pypi.org/project/pysbd/
17. Sentence segmentation and word tokenization (syntok) [internet]. Available from https://pypi.org/project/syntok/
18. spaCy Linguistic features [internet]. 2021. Available from https://spacy.io/usage/linguistic-features
19. Download MEDLINE/PubMed Data [internet]. Available from www.nlm.nih.gov/databases/download/pubmed_medline.html
20. Structured Abstracts in MEDLINE [internet]. Available from https://structuredabstracts.nlm.nih.gov/
21. Cutts, M. Oxford guide to plain English. 3rd ed.: Oxford University Press, New York; 2009.
22. English Letter Frequency Counts [internet]. 2012. Available from http://norvig.com/mayzner.html