NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

BSc THESIS

Techniques for sentence-boundary detection in Greek legal text

Ioannis B. Papastamou

Supervisors: **Manolis Koubarakis,** Professor
Despina - Athanasia Pantazi, PhD Candidate

ATHENS

March 2023

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Τεχνικές για Διαχωρισμό Ελληνικών Νομικών Κειμένων σε Προτάσεις

Ιωάννης Β. Παπαστάμου

Επιβλέποντες: **Μανόλης Κουμπαράκης,** Καθηγητής
Δέσποινα – Αθανασία Πανταζή, Υποψήφια Διδάκτωρ

ΑΘΗΝΑ

Μάρτιος 2023

BSc THESIS

Techniques for sentence-boundary detection in Greek legal text

Ioannis B. Papastamou

S.N.: 1115201400252

SUPERVISORS: **Manolis Koubarakis,** Professor
Despina - Athanasia Pantazi, PhD Candidate

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Τεχνικές για Διαχωρισμό Ελληνικών Νομικών Κειμένων σε Προτάσεις

Ιωάννης Β. Παπαστάμου

Α.Μ.: 1115201400252

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Μανόλης Κουμπαράκης,** Καθηγητής
Δέσποινα – Αθανασία Πανταζή, Υποψήφια Διδάκτωρ

ABSTRACT

Sentence Boundary Detection (SBD), also known as sentence boundary disambiguation, is a key underlying task for Natural Language Processing (NLP). Although SBD is considered to be a simple problem, it becomes more complex in other domains due to unorthodox use of punctuation symbols. For example, drug names in medical documents, case citations in legal text and references in academic articles, all use punctuation in ways which are uncommon in common documents such as the newswire documents. SBD is also a task that is language dependent. Every language brings its own unique problems when it comes to SBD. SBD has generally not received much attention in the field of the NLP research. The current thesis examines different ways SBD can be applied to the Raptarchis Dataset. We develop two SBD systems, each based on a different approach, and we analyze their advantages and disadvantages. We conclude, by using the SBD system that performed better, and provide a new version of the Raptarchis dataset with its sentences annotated.

SUBJECT AREA: Artificial Intelligence

KEYWORDS: Natural Language Processing, Legal Documents

ΠΕΡΙΛΗΨΗ

Η ανίχνευση ορίων προτάσεων (SBD) γνωστή και ως αποσαφήνιση ορίων προτάσεων, ή και πιο απλά Διαχωρισμός Προτάσεων, είναι μια βασική υποκείμενη εργασία για τον κλάδο της Επεξεργασία Φυσικής Γλώσσας (NLP). Αν και ο Διαχωρισμός Προτάσεων θεωρείται απλό πρόβλημα, γίνεται πιο περίπλοκο σε άλλους τομείς λόγω της ανορθόδοξης χρήσης των συμβόλων στίξης. Για παράδειγμα, τα ονόματα φαρμάκων σε ιατρικά έγγραφα, οι τίτλοι σε νομικά κείμενα και οι παραπομπές σε ακαδημαϊκά άρθρα χρησιμοποιούν τα σημεία στίξης με τρόπους που δεν είναι συνηθισμένοι όσο είναι οι τρόποι που χρησιμοποιούνται σε κοινά έγγραφα όπως στα έγγραφα ειδήσεων. Ο διαχωρισμός προτάσεων είναι επίσης μια εργασία που εξαρτάται από τη γλώσσα. Κάθε γλώσσα φέρνει τα δικά της μοναδικά προβλήματα όταν πρόκειται για το διαχωρισμό προτάσεων. Ο διαχωρισμός προτάσεων γενικά δεν έχει λάβει τόση μεγάλη προσοχή στον τομέα της έρευνας NLP. Η πτυχιακή αυτή εξετάζει διαφορετικούς τρόπους με τους οποίους ο διαχωρισμός προτάσεων μπορεί να εφαρμοστεί στο σύνολο δεδομένων Raptarchis. Αναπτύσσουμε δύο συστήματα Διαχωρισμού Προτάσεων, το καθένα με βάση διαφορετική προσέγγιση, αναλύοντας τα πλεονεκτήματα και τα μειονεκτήματά τους. Ολοκληρώνουμε, χρησιμοποιώντας το σύστημα που απέδωσε καλύτερα, και παρέχουμε μια νέα έκδοση του συνόλου δεδομένων Raptarchis με τις προτάσεις να έχουν χωριστεί .

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Τεχνητή Νοημοσύνη

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Επεξεργασία Φυσικής Γλώσσας, Νομικά Έγγραφαφα

Ευχαριστώ τους γονείς μου για την ανιδιοτελή τους αγάπη, δεν θα βρισκόμουν εδώ χωρίς αυτούς.

ACKNOWLEDGEMENTS

I would like to sincerely thank my supervisors, Manolis Koumparakis and Despina-Athanasia Pantazi for their invaluable support and guidance throughout the planning and development of this Thesis. I couldn't ask for better guides to introduce me to the world of research. Our collaboration was flawless, they gave me the freedom to explore this topic as I saw fit and at my own pace.

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Natural Language Processing (NLP) includes all the processes performed by computers working with natural language, like speech and written text. NLP manipulates textual data and the multiple steps of this process build a pipeline. Most pipelines begin with the fundamental task of identifying sentences, or as it more formally referred to as Sentence Boundary Detection(SBD). SBD at first seems like a trivial problem, you simply have a number of candidate boundary points (i.e ".", "!", "?") and need to decide whether an occurrence of these points in your text will be classified as a sentence ending marker or not. In reality, the use of these candidate points is ambiguous, because they can be used for purposes other than ending sentences. For example, periods can be found at the end of a word signaling that the word is an abbreviation, close to numbers when used as a decimal point or as an indicator of position in a sequence and many other examples.

While SBD is an important task, it has not been receiving the attention it deserves from the NLP community, for a variety of reasons [12]. Previous work in SBD is mostly restricted to the news domain and limited datasets such as the Wall Street Journal [8]. The problem with such work is that focusing on the news domain has some benefits for SBD because the text is of good quality when it comes to formatting, spelling, grammar and sentence construction making SBD less ambiguous. There are however many domains that do not share the same benefits. For example, the medical and legal domains are filled with abbreviations and use punctuation in ways that are uncommon for news articles, not to mention the fact that the underlying structure of the documents of said domains is sometimes loosely followed. For instance, in legal text a document may be missing the date a law was passed or have missing references. Besides the aforementioned difficulties, language also poses a problem for SBD, as each language comes with its own set of punctuation marks and has subtle differences that have an impact on SBD[1]. Some examples are provided below:

- The Greek language uses the English semicolon (;) as a question mark (?)
- In German and French there are significant differences when writing numbers. The period (.) is used as a thousand separator (English 1,000 turns into 1.000) and the comma (,) is used as a decimal point.

This demonstrates that SBD systems are genre- and domain-dependent, but also language-sensitive. This dependency also makes portability hard.

The Raptarchis Dataset is a novel dataset consisting of more than 47 thousand official, categorized Greek legislation resources [11]. In this thesis, we will examine the methods that have been applied for SBD in the literature, decide which approach is the best for splitting the Raptarchis Dataset into sentences, and at the end enhance the Dataset by providing an updated version of it with our sentence boundaries.

This thesis is divided in chapters, each one of which is listed below:

- In chapter 2, we take a look at the approaches taken to tackle the SBD problem, we examine approaches that focused on the legal domain, and approaches that were designed to work for a variety of domains.
- In chapter 3, we decide what are the best approaches we can take to split the Raptarchis Dataset into sentences given our limitations and we built two SBD systems.

- In chapter 4, we examine the performance of our two SBD systems, and we choose one to annotate our Dataset.
- In chapter 5, we elaborate on our conclusions and give our thoughts about SBD moving forward.

2. BACKGROUND AND RELATED WORK

In this section, we will provide an overview of the approaches used to tackle the SBD problem. Over the years, several methods have been used to solve SBD, all of whom have had their moment in the spotlight. These methods take one of 3 approaches:

1. Rule-based, using a combination of hand-crafted rules and fixed lists of lexical items in order to decide which punctuation marks in the text signal sentence boundaries. They can work on raw unannotated text.
2. Supervised, making use of annotated datasets which already have their sentence boundaries marked.
3. Unsupervised, which do not operate on annotated datasets nor require hand-written rules. They instead make use of information derived from the text.

We are also going to take a look at how some researchers segmentend a dataset of US adjudicatory decisions into sentences [3].

2.1 Ruled-based approaches

2.1.1 Document-Centered Approach

Mikheev [9] focused their efforts on the two major sources of ambiguity when it comes to SBD, those being abbreviations left of a potential boundary point, and proper names that appear to the right of a potential boundary point. The method they used relied upon a small set of rules to disambiguate sentence boundaries supplemented by support resources in the form of 4 word lists that can be easily derived from unlabeled text, which include:

1. a list of english common words,
2. a list of common words most frequently used in sentence starting positions,
3. a list of single word proper names, and finally,
4. and most importantly, a list of known abbreviations.

What makes their method distinct is the way they disambiguate whether or not a word token is actually a proper name or an abbreviation. Instead of looking at the word token's immediate local context, they look at the unambiguous usages of the word tokens in the entire document (hence the name Document-Centered Approach). For instance, given the sentence "The state of South Cal. decides that", we want to decide whether or not the "." after *Cal* is a sentence ending marker or is part of the abbreviation *Cal.*. Looking over the entire document to see if the word *Cal* is found in a different context without having a trailing period attached to it, we can lean more towards the idea that the period in question is actually part of an abbreviation, or vice versa, if we find *Cal* with a trailing period next to it, then we would lean more towards the ides that it is an abbreviation.

All in all, the method they presented showed promising reporting error rates depending on the quality of the information derived from the text, and varied between 0.01% and 2.0%

on the Brown Corpus [5], and 0.13% to 4.0% on the WSJ Corpus [8]. The real advantage of this approach is that it is domain independent, one can take the principles of this approach and apply them to a corpus of any domain since all the support resources needed can be generated automatically from the corpus, without human intervention. Moreover the authors investigated the portability of this method to other languages and obtained encouraging results on a corpus of news in Russian (this is a custom corpus they compiled from BBC news articles in Russian and is not publicly available). This seems to suggest that this method can be used for other european languages, although anyone wishing to use this method must take into account the uniqueness of their language and the way punctuation is used in the said language. This would make the DCA an independent semi-language, in a sense, which would be extremely helpful for low-resource domains in different languages, for which it would be hard to find annotated datasets with sentence boundaries.

2.1.2 PySBD

PySBD [14] is one of the most recent instances, where a rule-based approach has been applied to the SBD problem, despite the fact that these days Machine Learning approaches are at the forefront for any NLP task. Rule-based approaches are mostly rejected due to them needing a lot of effort to set up the rules that are going to be used, as well as the need for supporting resources around the rules like a potential abbreviation list. Furthermore, rule-based systems are considered to be non-robust in that they usually do not perform well outside the language and domain that they were developed on. However, the authors of this paper focus on the positive features of the rule-based systems, namely that unlike Machine Learning models, the errors produced by a rule-based SBD system are interpretable, the system does not rely on training data, and that the rules and support resources of such a system can, to a certain extent, be adjusted for a different language.

PySBD makes use of a set of rules, designed to cover sentence boundaries across a variety of domains. These rules are interpretable since each rule targets a specific kind of sentence boundary, and the rule set is easy to extend with new examples of particular sentence boundary markers. For their experiments the authors used a rule set for each of the 22 languages they covered. These rules are derived by considering possible sentence boundaries per language, as well as considering different domains. For example, the English rule set consists of 48 different rules that were derived from many domains to cover a variety of phenomena. They evaluated the performance of the English version of their system on their own *Golden Rules Dataset* and the GENIA corpus [6], reporting 97.92% and 97% accuracy respectively. When it comes to languages other than English, they tested on the OPUS-100 multilingual corpus [15] getting decent results, excluding certain languages like Polish and Burmese, citing the lack of language specific knowledge to form rules and abbreviation lists as the reason for the said performance.

2.2 Unsupervised approaches

2.2.1 Punkt

The most notable of the unsupervised approaches is Punkt [7] which is also used by the Python NLP library NLTK [4]. Punkt is based on the assumption that a large number

of ambiguities in the determination of sentence boundaries can be eliminated once abbreviations have been identified. The authors note that abbreviations can be defined as collocations consisting of truncated words and periods. To detect the abbreviations, the system collects statistics about occurrences of tokens, punctuations, token length, casing, and also the collocation bond between tokens. It calculates probabilities based on these statistics and uses them in testing heuristics. These heuristics are basic rules that are used to decide if a token is, e.g., a frequent sentence starter, or build a collocation with a period. The results are used to classify a token as a sentence boundary. For the learning process, it requires only a larger amount of unlabelled text from the same domain as the target text. Their reported rates of errors on 'classic' test sets are 1.02% (Brown) and 1.65% (WSJ).

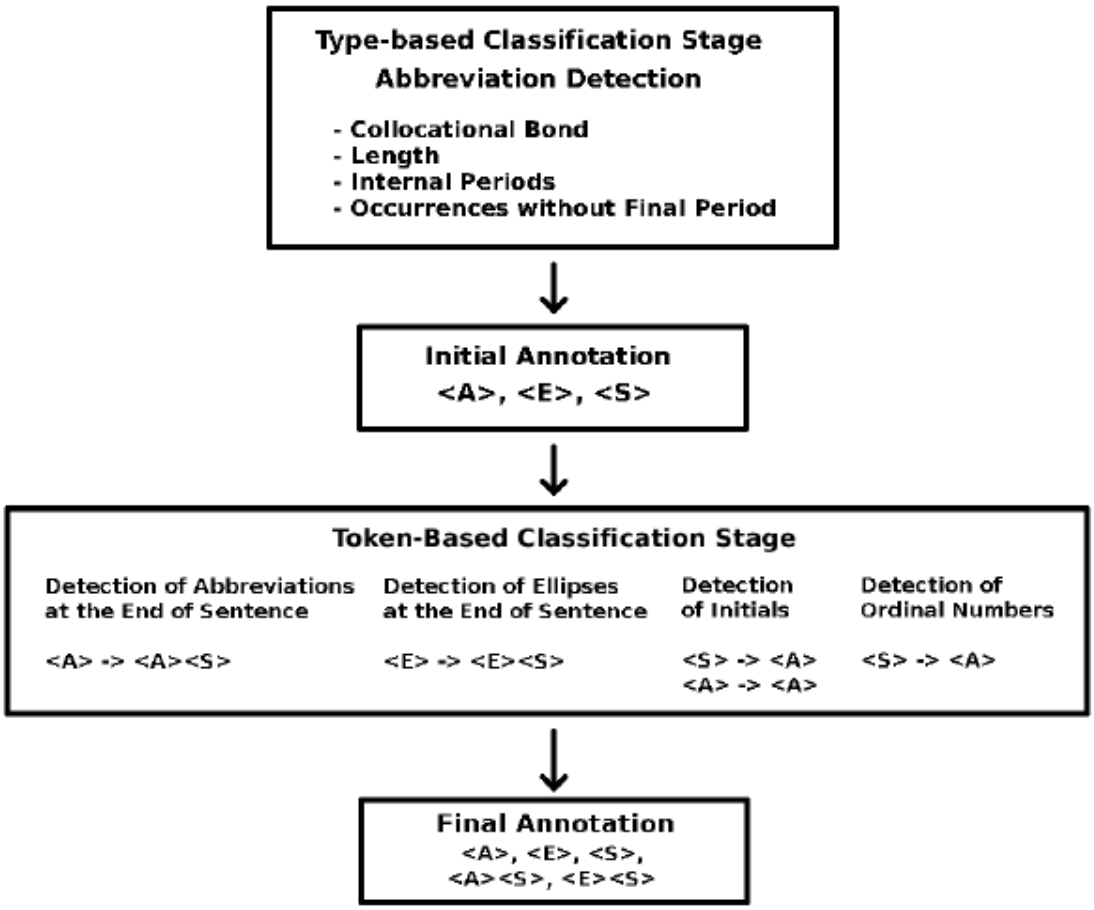

Figure 2.1: Architecture of the Punkt System

Each corpus goes through certain stages in the Punkt system. In the first stage, a resolution is performed on the type level to detect abbreviation types and ordinary word types. After this stage, the corpus receives an intermediate annotation where all instances of abbreviations detected by the first stage are marked with the tag <A> and all ellipses with the tag <E>. All periods following non-abbreviations are assumed to be sentence boundary markers and receive the annotation <S> . The second, token-based stage employs additional heuristics on the basis of the intermediate annotation to refine and correct the outputs of the first classification for each token. The token-based classifier is particularly suited to determine abbreviations and ellipses at the end of the sentence giving them the final annotation <A><S> or <E><S>. But it is also used to correct the intermediate annotation by detecting initials and ordinal numbers that cannot easily be recognized with

type-based methods and thus often receive the wrong annotation before the first stage.

2.3 Supervised approaches

2.3.1 SATZ

The SATZ system [10] makes use of a multi-stage architecture that ends with the classification of tokens as sentence boundaries by a neural network. The architecture of SATZ is shown in figure 2.2. The method starts with tokenizing the input text, which is done with their own custom tokenizer. After that, the system looks at the context preceding and following a punctuation mark, and uses probabilities of all parts-of-speech to tag this word. The system uses a lexicon that contains part-of-speech frequency data for each word with which it calculates the probabilities. In the case where the word is not present in the lexicon, the system uses heuristics to assign the most likely part-of-speech tag for that word, following that a vector of probabilities is constructed to describe each token, which is then fed as input to a feed-forward neural network. The SATZ system recorded slightly worse error rates than the previous attempt at a supervised model at the time [13]. More specifically, the error-rate increased from 1.1% to 3.3% on the WSJ corpus, but overall, the system benefits from being more robust and portable to new languages. The authors adjusted the lexicon and the heuristics to use the model in other languages such as French and German and got encouraging results.

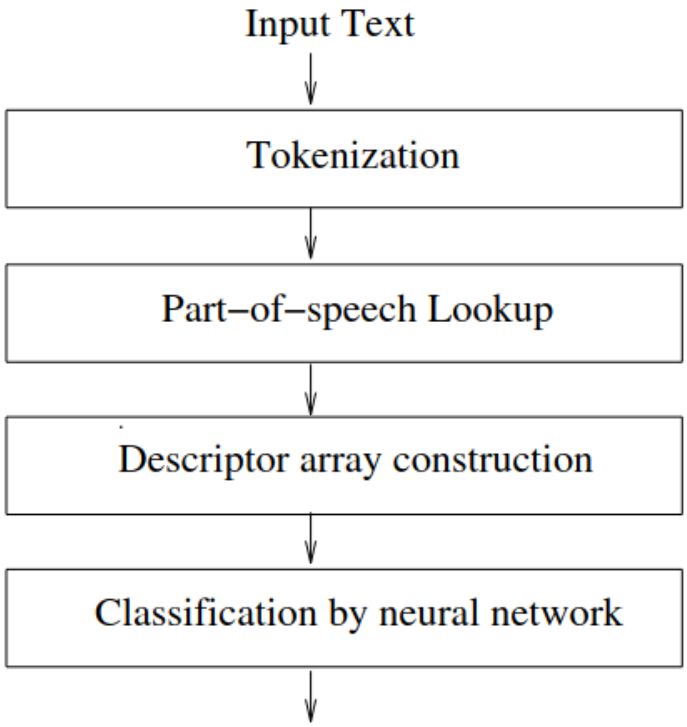

Figure 2.2: Architecture of the SATZ System

While SATZ system is a supervised model that makes use of a labeled dataset, it still relies on some external resources, mainly when it comes to the lexicon and the heuristics that are used to assign probabilities to each token, so, despite the fact that SATZ is supervised, it still needs some adjustments to work for other domains and languages.

2.4 Dataset Annotation

2.4.1 Savelka

While supervised systems are a great choice when it comes to applying an SBD method, they have one distinct drawback, and that is that in order to apply a supervised method you are in need of a dataset with its sentences annotated, something that is hard to come by when you want to apply a supervised method in a low resource language or domain.

Since our SBD system system is going to be developed for the legal domain, we will take a look at the way a dataset of US adjudicatory decisions was annotated [3], and see what we can take away for our endeavors. The authors adopted the protocol for sentence annotation developed by Research Laboratory for Law, Logic and Technology (LLT Lab) at the Maurice A. Deane School of Law at Hofstra University [2]. Such annotation protocols provide methods and criteria for manually annotating texts, and a set of conventions governing the generation of annotation data. Protocols are developed in two stages. First, from a sample of documents containing a variety of decisions, examples are collected that display normal forms of the annotation type, linguistic variants of those normal forms, and aberrant forms. Second, those examples are used to derive general guidelines, criteria and conventions for manually annotating these types within texts.

For the annotation type “Sentence”, the normal form is a grammatical subject con- sisting of a noun phrase followed immediately by a grammatical predicate consisting of a verb phrase - i.e., <grammatical subject noun phrase><grammatical predicate verb phrase>. Example normal form:

The Veteran’s chronic adjustment disorder with depressed and anxious features is related to service.

A span of text that is a “linguistic transform” of a normal form is one for which also constitutes an annotation of the type “Sentence”. Finally, there are spans of text that have a very particular linguistic structure (being neither normal form nor linguistic transform), but still constitute an instance of the type “Sentence”. We introduce all these spans of texts found in legal documents , that constitute an annotation of type “Sentence”.

The title of a legal document should be considered a single sentence. Titles usually have a distinct format, express a single thought and they can span multiple lines of text. Headings are also annotated as sentences. They are used to determine the structure of the text and the relation between the different segments of the document, which in term gives us information about the overall structure of the text (i.e for this thesis INTRODUCTION is considered a heading).

Ellipses usually occur in legal documents in one of two ways. Firstly, they can appear within a sentence indicating missing words from within the sentence, in which case the ellipses are included within the overall sentence span. Secondly, they can occur between sentences, which means that they should be annotated as separate sentences. This way of using ellipses gives coded information to the reader, like that sentences have been deleted from this passage and ellipses are used in their place.

Parentheticals inside sentences are widely used in the legal domain. The authors chose an approach to always annotate parentheticals within the span of an overall sentence. This in term means that even if the parentheticals contain other sentences inside them.

These sentences will never be separately annotated. For example, the following is a single sentence:

Id. at 576, 128 S. Ct. 558; see also id. at 575, 128 S. Ct. 558 ("The District Court began by properly calculating and considering the advisory Guidelines range. It then addressed the relevant § 3553(a) factors.")

Colons as sentence-ending punctuation can sometimes occur as an exception to the normal presumption that a colon is not sentence-ending punctuation. This mostly happens when the colon is used as the last punctuation mark in a paragraph and is immediately followed by a line break. For the most part the use of colons is stylistic. The author makes the choice of using the colon instead of the period to express that what comes after the colon is related to what came before. Moreover, the colon followed immediately by a line break is used to introduce a block quote or an enumerated list of items, thus making the colon a good place to end a sentence. The block quote and enumerated lists can later be annotated into separate sentences themselves.

Enumerated lists are widely used in legal documents and they can be both numbered and lettered. Their treatment largely depends on whether or not the list items themselves are sentences or not. If the list item is a sentence then the list item is annotated as a stand alone sentence without the list number. For example:

1. This is a sentence.

The above example contains two sentences, one being the "1." and the other being "This is a sentence". This way of thinking about annotating enumerated lists helps in the case of numbered lists. The authors note that this is better for Machine Learning, as the number introducing the list item should always be its own sentence. If the list items are not themselves sentences, then there is one overall sentence that includes the list items, and the list numbers or letters occur within that overall sentence. In such a case, there is only one sentence. For example, the following is a single sentence containing an enumerated list:

Supermarket list: 1) eggs 2) milk 3) ham 4) chicken.

3. APPLYING SBD ON RAPTARCHIS

In this section we are faced with the task of applying SBD on the Raptarchis Dataset.

3.1 About Raptarchis

The “Permanent Greek Legislation Code - Raptrachis3” contains Greek legislation until 2015, since the creation of the Greek state in 1834. It includes laws, decrees, regulations and decisions with their respective amendments such as replacements, modifications and deletions, while its only source of information is the Official Government Gazette. It consists of 47 legislative volumes and each volume corresponds to a main thematic topic. Each volume is divided into thematic subcategories which are called chapters and subsequently, each chapter breaks down to subjects which contain the legal resources. The total number of chapters is 389 while the total number of subjects is 2285. Each legal document resource is a json file whose structure is best explained by figure 3.2. It contains several fields, but for our purposes we only use the fields title, header and articles.

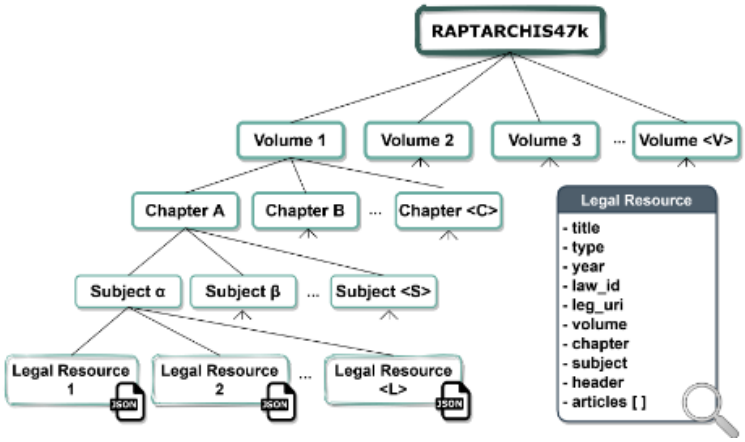

Figure 3.1: Raptarchis dataset structure


```
{
  "title": "17. ΠΡΟΕΔΡΙΚΟ ΔΙΑΤΑΓΜΑ υπ΄αριθ. 234",
  "type": "ΠΡΟΕΔΡΙΚΟ ΔΙΑΤΑΓΜΑ",
  "year": "1996",
  "law_id": "234",
  "leg_uri": "http://legislation.di.uoa.gr/eli/pd/1996/234",
  "volume": "ΒΙΟΜΗΧΑΝΙΚΗ ΝΟΜΟΘΕΣΙΑ",
  "chapter": "ΔΙΑΦΟΡΕΣ ΒΙΟΜΗΧΑΝΙΕΣ",
  "subject": "ΔΗΜΟΣΙΑ ΕΠΙΧΕΙΡΗΣΗ ΠΕΤΡΕΛΑΙΟΥ",
  "header": "Αύξηση κατά δεκαπέντε [...] Α.Ε.».",
  "articles": [
    "\nΕγκρίνεται η από 22 [...] δραχμών η κάθε μία.",
    "\nΤο Διάταγμα αυτό ισχύει από την [...] διατάγματος."
  ]
}
```


Figure 3.2: Raptarchis dataset legal document-resource example

3.1.1 Sentence Boundaries on Raptarchis

Since we are in the legal domain and in the Greek language we want to define which characters we will consider as candidate sentence boundaries. Below is a table of characters we might want to consider as Sentence Boundaries along with their counts.

Table 3.1: RAPTARCHIS Candidate Sentence Boundaries

Name	Symbol	Count
Fullstop/Period	.	1839432
Colon	:	9511
Exclamation Mark	!	29
Question Mark	?	68
Semicolon/Greek Question Mark	;	288

As expected, the period character '.' is by far the most common character, makes sense since it is the main indicator of end of sentence and should always be considered as a sentence boundary indicator, colon ':' also seems to be common enough that it makes sense to be included as a candidate sentence boundary (it is used to introduce enumerated lists more on that later). As for the rest of the characters, their usage is way too low to warrant consideration, and when taking a look into how they are utilized in the dataset, they are almost never used to end sentences.

3.1.2 Abbreviations in Raptarchis

When it comes to finding sentence boundaries, abbreviations are a major source of ambiguity, the types of abbreviations found in Raptarchis are the following:

- Type 1: "Ο.Α.Ε.Δ." - "κ.λ.π."
  Uppercase-lowercase letters that are separated by periods and end on a period.
- Type 2: "ΝΟΜΟΘΕΤ." - "Απόφ." - "απόφ."
  Uppercase-lowercase letters that end in a single period.
- Type 3: "ΕΜΠ" - "BBC"
  Uppercase letters that are not separated by periods.

Abbreviations introduce ambiguities for sentence ending marking simply by ending on periods, this means that we cannot be certain that a period ends a sentence if it is part of an abbreviation, moreover abbreviations do not form a closed set, that is one cannot list all possible abbreviations. The identification of abbreviations is gonna play a major role in developing an SBD system and is something that we will deal with in the following Sections.

3.1.3 Capitalized Words/Proper Names in Raptarchis

Like abbreviations, capitalized words are a source of ambiguity when it comes to detecting Sentence Boundaries. In mixed-case texts like Raptarchis, capitalized words usually denote proper names (names of organizations, people, locations etc.), but there are special

positions in the text where capitalization is expected. Such mandatory positions include the first word in a sentence, words in titles with all significant words capitalized and the first word in a list entry among others. Capitalized words in these and some other positions present a case of ambiguity, since they can stand for proper names, or they can just be capitalized common words. The disambiguation of capitalized words in ambiguous positions leads to the identification of proper names. Disambiguating capitalized words in Raptarchis is important to us since if we know that a capitalized word that follows a candidate sentence boundary is a proper name, we can make a better decision on whether or not that candidate sentence boundary should end a sentence.

3.1.4 Similarities between Raptarchis and U.S. legal decisions

What we took away from the Savelka paper [3] is that there are certain spans of text found in legal documents that could constitute sentences in U.S. legal documents, moving forward we would like to identify which of these sentence types appear in Raptarchis.

3.1.4.1 Case names in Document titles

Raptarchis documents do have titles. In particular the title can be found in the “title” field and is most of the time also included in the header. Titles in Raptarchis follow a consistent format, they start with a number, a sequence of capitalized letters, and then a number after that (an example of a title can be seen in figure 3.2).

3.1.4.2 Enumerated Lists

Lists, whether we are talking about numbered or lettered lists, appear commonly in Raptarchis. Enumerated lists are the reason we include the colon in our candidate sentence boundaries, since almost always enumerated lists in Raptarchis are introduced after a colon.

Figure 3.3 shows an example of two enumerated lists, one is numbered the other is lettered, there are also other ways enumerated lists are introduced, but in general enumerated list entries follow these rules.

- The first character of each entry will be a number or a letter immediately followed by either a period ‘.’ or a right parenthesis ‘)’.
- When it comes to letters those can be uppercase or lowercase greek letters like A), α) or B), β) respectively.
- Also there are times where Latin numbers are used in the place of regular numbers for example list entries will be introduced with either the uppercase version of I) II) III) IV) … or the lowercase version i) ii) iii) iv) and so on.

Ideally we would like to be able to identify enumerated lists of all types so that we could split them into sentences.

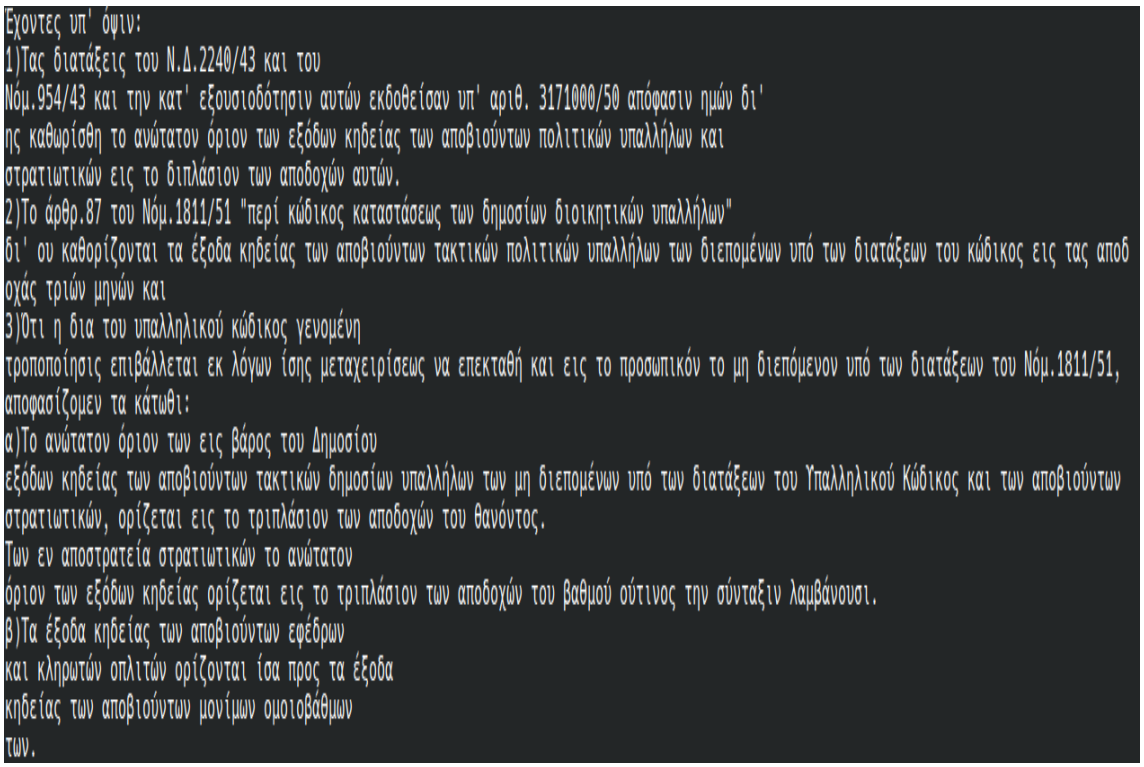

Figure 3.3: A Raptarchis enumerated list

3.1.4.3 Typos/Grammatical Errors/Missing text

While most of the documents in our dataset are of decent to good quality, there are many documents that have typos like missing the date in which the document was written, list entries not having periods, at the point where the list entry ends and a new one begins, and documents not having correct spacing between words, which leads to a slew of other problems like misrecognition of abbreviations and proper names. Errors like these propagate to other stages of the NLP pipeline, but most importantly, they make our efforts of disambiguating sentence boundaries harder, because we cannot always rely on our input texts being of good quality.

3.2 Building a SBD system for Raptarchis

As we discussed in Section 2, there are 3 approaches for SBD, supervised, unsupervised, and rule-based. Any supervised SBD method would need a dataset with its sentence boundaries already marked. To our knowledge there is no dataset in the Greek Legal Domain with its sentences already annotated that could be used to train a model which would be subsequently be used to split Raptarchis into sentences. This means that a supervised approach is off of the table. This fact leaves us with the other 2 options. In the following subsections we will describe how we built 2 SBD systems, one will be a rule-based statistical system mainly drawing inspiration from Mikheev [9] and using information about how the authors of [3] tackled the problems with SBD when it comes to the legal domain, and the other will be an unsupervised model based on the algorithm presented in Punkt [7] which is provided with the Python library NLTK [4].

3.2.1 Rule-based Approach

If we had at our disposal entirely correct information about the words that appear in the same local context as our potential boundary points, deciding on if those potential boundary points are indeed sentence ending would be easy because we would know every time whether or not a word left of a potential boundary point is an abbreviation and a word right to the right of it is a proper name.

Our rule-based system will mainly rely on three things when it comes to deciding whether or not a candidate boundary point is in fact a sentence boundary, first is a list of support resources that we will derive from our dataset that help us decide if certain words in ambiguous contexts should end/start sentences, second is an abbreviation guessing heuristic that helps us identify abbreviations left of candidate boundary points and third is all those decisions we make about words that can be Sentence Starters.

3.2.1.1 Setting up Our method

In this subsection we will describe all the things we do before applying the main heuristic that splits our text into sentences.

3.2.1.2 Boundaries Tagging

Raptarchis documents are given to us in the form of json files, out of each json file we concatenate the title (if it is not included in the header) the header and the articles, forming a single string. To mark which characters of that string are sentence ending we will tag each character with a label. This label will also denote if the character in the text starts a sentence (label ‘B’), is included in a sentence (label ‘I’) and ends a sentence (label ‘L’). Below is a list of things we gain by using this supplementary tagging method.

- Firstly, we make it possible for sentences to be included into other sentences, meaning that we can start from 1 sentence that spans our text, and then further segment it into more sentences by updating the labels.
- We can initialize the end of the header and the end of each article with a ‘L’ label already segmenting these parts of our text as sentences.
- We do not allow sentences to be less than two characters long, so a ‘B’ label needs to be followed by zero or more ‘I’ labels and then one ‘L’ label, in other words we cannot have consecutive ‘B’s and ‘L’s.
- We do not allow the update of labels that are not ‘I’, this means that once a label is tagged with ‘B’ or ‘L’ it cannot be changed.
- When our heuristics find that a certain character is sentence ending we update its label to ’L’ and the label of the next character to ’B’. This is done to maintain a balance between the ‘B’ labels and the ‘L’ labels.
- We can always check if our segmentation is valid by checking whether or not the number of ‘B’ labels is equal to the number of ‘L’ labels.

- When all our heuristics are applied we can simply refer to the tags of each character to provide the text splitted into sentences.

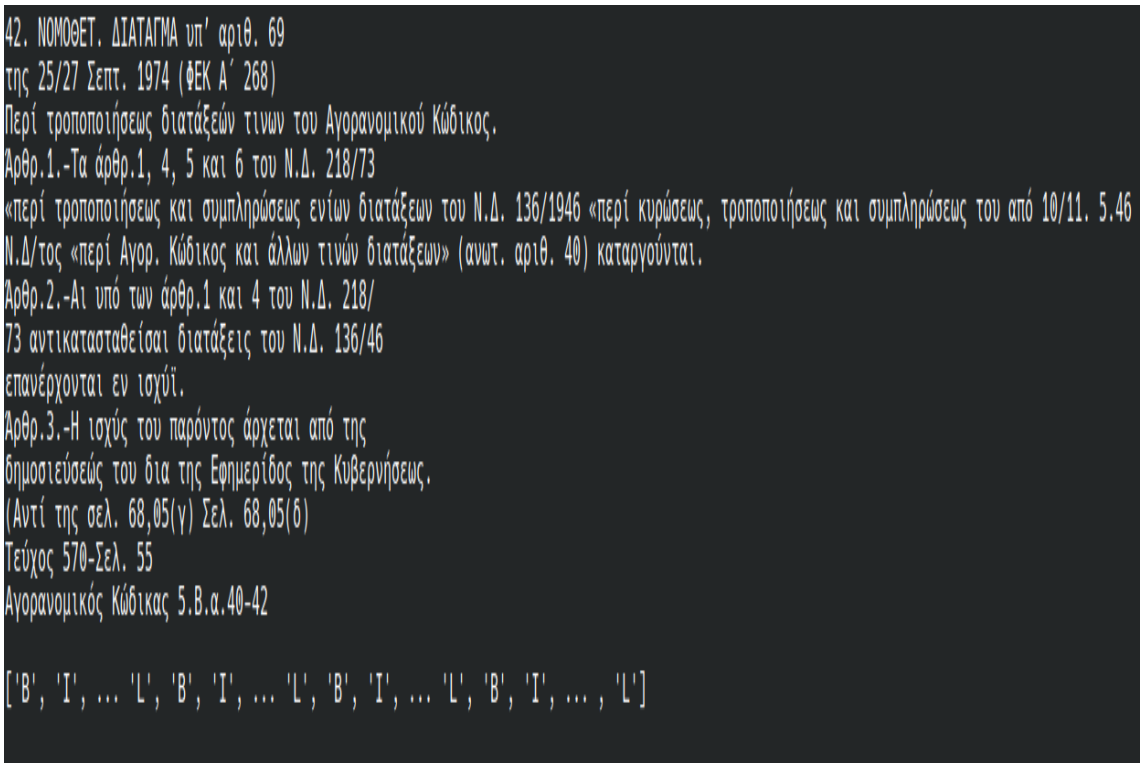

Figure 3.4: Example of an input text with a header and 3 articles before applying SBD

3.2.1.3 Building Support Lists

Our rule-based approach will also make use of four word lists. Each list is a collection of words that belong to a single type. This fact does not limit a word from appearing in more than one list, four lists means four types those being:

- common word
- common word that is a frequent sentence starter
- single word proper name
- abbreviation

These four lists will be acquired by making use of the raw text provided by our dataset more or less in the same way the authors of this paper did [9]. This also helps our lists to be of better quality since we are getting from the same dataset that we are gonna apply them to, as opposed to using another dataset in the Greek legal domain.

Getting a list of common words is simple, we simply count the times we found a word lower-cased in the text, and then include in the common-words-list those that were found at least 3 times. The list of common words that we ended up with had close to 123.000 Greek words.

The frequent starters list is a list of common words that are most frequently used in sentence starting positions. The problem here is that again we have no dataset with its sentence boundaries marked, so the question is how can we get a list of greek common words that are used as sentence starters. The authors of [9] compile that list by tagging capitalized words in sentence-starting as sentence-starters if they were found in the list of common words. To get a good quality list, one must know that the words that are considered for the frequent starters-list are in positions that can start sentences. To get the best quality list we can for our dataset, we are going to get a bit more creative. As mentioned before typos can be found in Raptarchis. Looking into the text, one can find improper spacing, missing characters amongst other things, but the good thing is that the majority of examples have good formatting. In other words, there are more “good apples” than “bad apples”, so we want to target these examples that have good formatting.

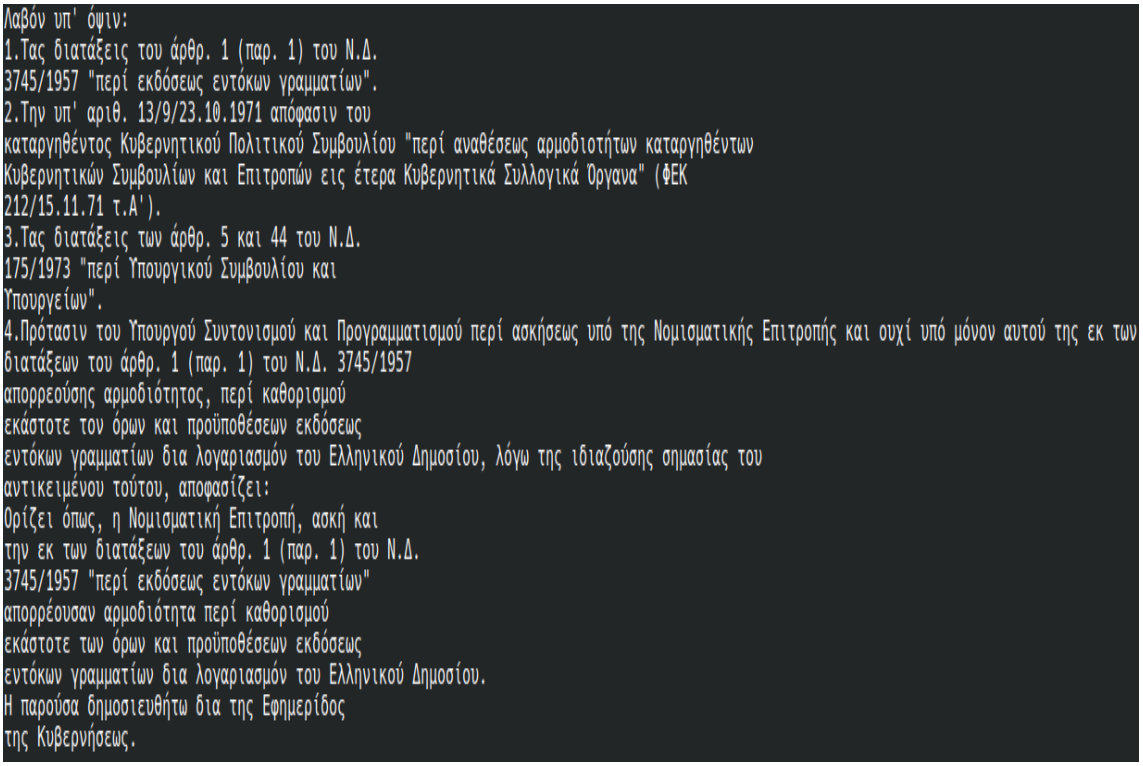

Figure 3.5: Text in Raptarchis that is well formatted

The Figure 3.5 shows what we consider good formatting. Every candidate sentence boundary point for our task is immediately followed by a line break, so the author of this has already in a sense given us the sentence boundaries. Applying a simple heuristic of marking a sentence boundary, whenever we encounter a candidate boundary point, immediately followed by a line break, while making sure that word left of it is not an abbreviation would give us the correct sentence splitting on this piece of text. So, for compiling our frequent-starters list, we want to make use of examples like this. We are going to use the Python string split method to split on line breaks (“newline character”) and then on spaces. This gives us the text in a list of strings, which we iterate over. For every step of the iteration, we are going to maintain in a variable the last word of the previous line. When that word ends on one of our candidate sentence boundary points and is not an abbreviation (we check this by using the abbreviation list we already 3.6 have in conjunction with our abbreviation heuristic 3.2.1.4) we know that we can look at the first word of the current line, to add that word in our frequent-starters list. The following two things must be true for the word to be added to our frequent-starters list:

1. The word must not be fully capitalized(i.e EXAMPLE) because we treat fully capitalized words as proper names and proper names do not start sentences by default.

2. The word needs to start with a digit followed by a '.' or a ')' and an uppercase letter or the word needs to start with an uppercase letter, while in both cases the word must be in the common words set.

We count the words that fit the above criteria, and the 200 most frequent of them are added to our frequent-starters list.

The single-word-proper-name list includes the 200 words that were most frequently seen as single capitalized in unambiguous positions, and at the same time, were present in the common words list. What we deem as unambiguous positions for proper names are those where the word in question is found capitalized while being preceded and followed by lowercase words.

Finally, for our fourth and final list, we compiled statistics about which words appear left of periods and went through the statistics to see which ones were actually abbreviations. This task requires human effort because recognizing abbreviations of type 2 like "απόφ." cannot be done automatically since any sequence of letters followed by a period could be an abbreviation. This goes back to what we mentioned earlier about abbreviations not forming a closed set. We also added to the list all uppercase and lowercase letters of the Greek alphabet, since it is clear to us that a sentence should never end on a single letter followed by a candidate boundary point. This process leaves us with a list of 224 abbreviations.

```
abbreviations = {"Ιαν", "Φεβρ", "Φεβ", "Μαρτ", "Απρ", "Απριλ", "Μαι", "Ιουν", "Ιουλ", "Αυγ", "Σεπτ", "Οκτ", "Νοεμ", "Νοεμβρ", "Δεκ",
    "απόφ", "Απόφ", "κατωτ", "νομ", "Νόμ", "Νομ", "αρθρ", "άρθρ", "Αριθ", "Αρίθ", "αριθ", "αρίθ", "παρ", "δρχ", "δραχ", "Δραχ",
    "Δρχ", "τόμ", "τομ", "ανωτ", "Γεν", "Δημ", "Σελ", "σελ", "διόρθ", "σφαλμ", "κλπ", "Επικ", "Οικ", "Εθν",
    "Διόρθ", "Σφαλμ", "Μιχ", "ΝΟΜΟΘΕΤ", "Εμπορ", "κατώτ", "ΥΠΟΥΡΓ", "Συνεδρ",
    "Κοιν", "ΚΟΙΝ", "Υπ", "εδ", "κλ", "κτλ", "Αν", "Εμπ", "Υγ", "Μεγ", "Βορ", "Ηνωμ", "Κυβερ", "Κωδ", "Κωδικ", "Εργ", "Συλ",
    "Πανελ", "Αναπλ", "Ημαρτ", "Επικοιν", "Αβ", "Πρ", "Ελλ", "Ανων",
    "αρ", "Οικον", "οικον", "περιπτ", "Ταχ", "Μεταφ", "ΚΟΙΝΩΝ", "Προεδρ", "ΠΡΟΕΔΡ", "Λιμ", "ΕΘΝ", "Τόμ", "Γραμ", "οικ", "ΟΙΚ"
    "πράξ", "πραξ", "χιλ", "αποφ", "υπ'αριθ", "Πράξ", "Πραξ", "τευχ", "τεύχ", "Τευχ", "Τεύχ", "ΕΜΠΟΡ", "Κυβ", "Κυβερν",
    "βλ", "Βλ", "απ", "κοχ", "Συμβ", "Εποπτ", "αγροτ", "Ενεργ", "Βιομ", "Στρατ", "Ποιν", "Συντ", "Ναυτ", "τροποπ", "συμπλ",
    "εδαφ", "παραγρ" , "Εσωτ", "Περιφ", "Εργατ", "Επαγγ", "ολομ", "Ασφαλ", "Υπηρ", "αριθμ", "αιρθ", "γραμμ", "διδασκ",
    "ημερ", "τεχν", "χλμ", "Εργασ", "κυβ", "τετρ", "εκατ", "ασφ", "στοιχ", "γραμ",
    "Δικον", "Πολιτ", "Εισαγ", "Κώδ", "Κωδ", "Παρασκ", "Βοηθ", "Επιμελ", "Ασφ", "Στοιχ", "Γραμμ",
    "Γεν", "Νοσ", "Θωρ", "Περ", "Παρ",
    "γεν", "περ", "ποιν",
    "ΥΕΘΑ/Γ.Ε.Α", "Υ.ΕΘ.Α./Γ.Ε.Α","Ν.Α.Τ.-Κ.Π.Φ.Ν",
    "Χρ", "Χαρ", "Νικ", "Νικήτ", "Ιακ", "Στυλ", "Παν", "Ειρ", "Αγγ", "Δημ", "Αναστ", "Αχιλ", "Βασ",
    "α", "β", "γ", "δ", "ε", "ζ", "η", "θ", "ι", "κ", "λ", "μ", "ν", "ξ", "ο", "π", "ρ", "σ", "τ", "υ", "φ", "χ", "ψ", "ω",
    "Α", "Β", "Γ", "Δ", "Ε", "Ζ", "Η", "Θ", "Ι", "Κ", "Λ", "Μ", "Ν", "Ξ", "Ο", "Π", "Ρ", "Σ", "Τ", "Υ", "Φ", "Χ", "Ψ", "Ω"}
```

Figure 3.6: Our abbreviation list

3.2.1.4 Abbreviation guessing heuristic

As we mentioned earlier, we recognize 3 types of abbreviations in the Raptarchis dataset 3.1.2. It is very difficult for a guessing heuristic to recognize type 2 abbreviations because virtually any sequence of letters that ends with a period can be an abbreviation. This is also a good time to mention that type 3 abbreviations do not impact SBD since they don't end on periods. The guessing heuristic we developed is 100% accurate when it comes to identifying type 1 abbreviations like "Ο.Α.Ε.Δ.", and will decide whether or not a word left of a potential boundary point is an abbreviation. This means that if the word given to the heuristic is an abbreviation it will be missing its last period. Our checks go as follows:

1. If the word ends on a period, it is not considered an abbreviation, since as we mentioned we get words to the left of periods. This also means that we treat ellipses as non abbreviations.
2. If the word is only made up of letters, it is not recognized as an abbreviation. This is done so that common words left of a boundary point are recognized as non-abbreviations.
3. And lastly, if the word contains characters other than periods and letters, it is recognized as a non-abbreviation.

When a word passes all these checks it is considered an abbreviation by our heuristic. If we look at our heuristic's rules, we can see that there are some strings that are non-abbreviations that still pass from all the checks. For example, a string that is made up of letters and periods like ".Ε.Δ" will be recognized as an abbreviation despite the fact abbreviations can't start with periods, the reason we allow this is because due to improper spacing in some Raptarchis documents, there are abbreviations that are splitted by a space character, allowing string like ".Ε.Δ" to be recognized as abbreviations. This allows our heuristic to perform better on our dataset. The fact that our heuristic can recognize all abbreviations of type 1 is the reason that there are no abbreviations of type 1 in our abbreviation list 3.6.

3.2.1.5 Splitting Raptarchis into Sentences with our Rule-Based Method

At this point we have everything prepared to start applying our heuristics for splitting each document into a sentence. We load our support lists and then we load each document one by one. We make a boundaries tagging representation for the current document, and then apply our main heuristic for sentence splitting.

3.2.1.6 Marking the Title

Before applying our main heuristic we are gonna first try to make more use of the 'title' field of the document. As described in Savelka, the title should be considered its own sentence. With Raptarchis we have access to the title field so we can always mark the title as its own sentence, but if we look more closely to the 'header' of the document, we can make the case that the actual 'title' of the document continues into the header, so we will try for each document to find the limits of the expanded title in the header and mark it as a sentence.

title : “8. ΒΑΣΙΛΙΚΟΝ ΔΙΑΤΑΓΜΑ”
header : “8. ΒΑΣΙΛΙΚΟΝ ΔΙΑΤΑΓΜΑ της 18 Ιαν./3 Φεβρ. 1954 (ΦΕΚ Α΄ 20)”

As the above example shows the expanded title usually has a date associated with it and ends on code which in the example is denoted by “ΦΕΚ Α΄20”, to find the limits of the expanded title we are going to search for the string “ΦΕΚ” in the header. If that string is found early in the header, then we mark the first line break after it as a sentence boundary. If the string “ΦΕΚ” is not found or is found far from the ‘title’ then we default to using the ‘title’, as a our first sentence.

3.2.1.7 Main Heuristic

For each text, we start by searching for occurrences of these strings {“[period or colon][line break]”, “[period or colon][space][line break], [period or colon][space]”}. Whenever we match one of these strings in our text, we examine its local context to see if we will end a sentence.

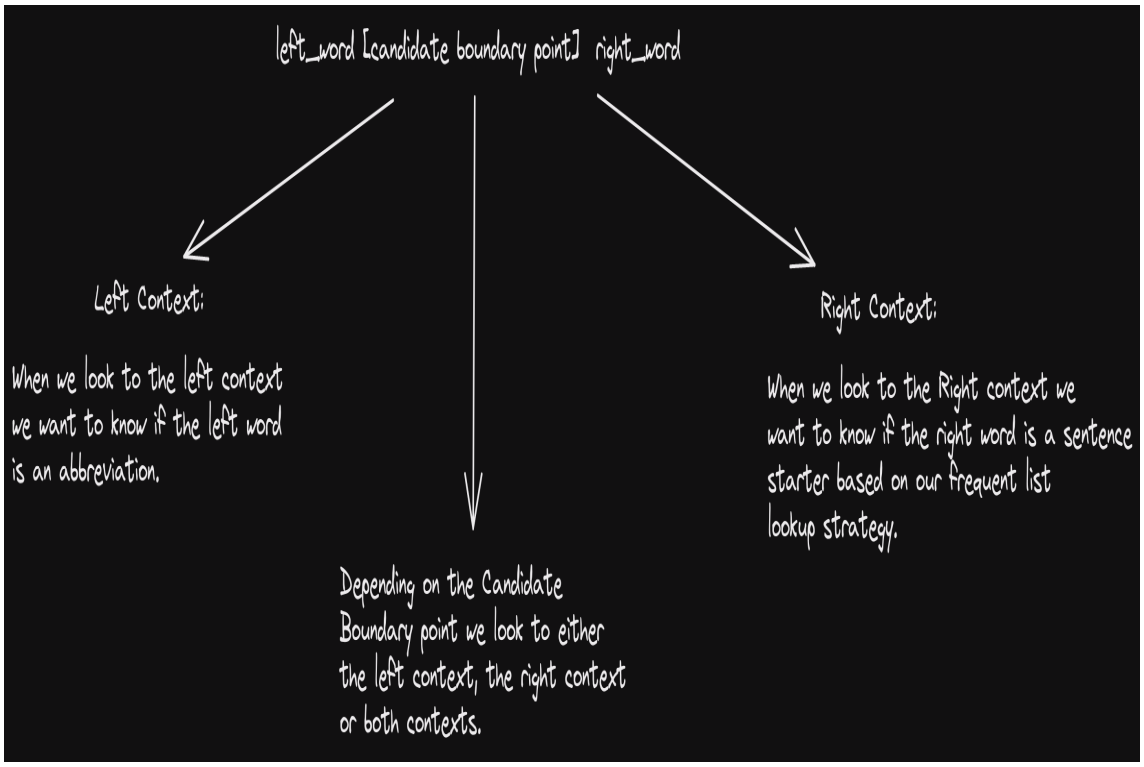

Figure 3.7: All cases encountered by our main heuristic

When the candidate boundary point in Figure 3.7 is a colon (‘:’) we only look at the right word. If that right word is found to be a sentence-starter, then we mark that colon as a sentence boundary. In the case where we matched a colon, and then a line break immediately after it, we don’t even make use of the right word, because we consider the colon followed by a line break (or space and then line break) a heavy indicator that an enumerated list follows the line break. The reason we only look at the right context is because even if an abbreviation comes before the colon, it has no impact on whether or not that colon is sentence ending.

When the candidate boundary point in Figure 3.7 is a period (‘.’), we will make use of both the left and the right word. We again consider line breaks as a strong indicator of new

sentence introduction, so whenever our period is followed by a line break (or space and then line break) we are going to look only to the left context. If the left word is not in our abbreviations list and our abbreviation guessing heuristic recognizes it as non-abbreviation, we mark the line break as a sentence boundary. Only when the word is in fact an abbreviation will we look to the right context to see if the right word can start a sentence. In the case of the candidate boundary point being a period only followed by space, we are going to look only to the right context to see if the right word starts a sentence. The reason we only look to the right in this case is because even if the word to the left is an abbreviation we still have to decide if the period we matched is sentence ending and in this case there is no line break to help us decide, so looking to the right word is necessary.

Deciding if a right word is a sentence-starter When given with a word to the right of a potential boundary point, we have to decide if that word is sentence starting. We are going to make use of the frequent-starters-list and our single-word-proper-name-list, but first every word that is found to be fully capitalized (i.e “EXAMPLE”) is considered a proper name by default, and therefore cannot start a sentence, and second if the word to the right starts with a digit or letter (uppercase or lowercase) immediately followed by a ‘)’, we recognize the right word as sentence starting. This second rule secures that we can recognize enumerated lists that use right parentheses ‘)’ to introduce list items (the reason this rule does not work for enumerated lists that use periods ‘.’ instead of right parenthesis because it missclassifies some right words as sentence starting when they are not). Finally, we need to do something about the case where the right word is found on both the frequent-starter and single-word-proper-name lists. This is truly an ambiguous case which we choose to resolve by classifying the word as a proper-name. On the other hand, if the right word is found on only one of the two lists, the corresponding list-type is returned. Additionally, after examining the words that appeared on both lists, we found that these words were some of the most frequent sentence starters and some of the most common sentence starters of the Greek language in general. These words were usually no more than 3 letters long. Examples include, ‘Οι, ‘Το’, ‘Για, ‘Με’, the equivalent scenario in English would be finding a word like ‘The’ in the single-word-proper-name-list. To reduce the number of words that appear in both lists we substitute such words with the next higher counted single word proper names to still have a list that has 200 words. In total we reduced the number of words that appear on both lists to only three.

3.2.1.8 Parentheticals within Sentences

One last thing is that sentence boundaries are only marked when the candidate sentence boundary point exists outside of parentheses. We take this directly from the Savelka paper [3] where the authors chose to never annotate sentences that were inside parentheses, choosing to always treat them as part of the overall sentence that the parenthese are in.

3.2.2 Punkt Approach

Punkt is an unsupervised model that is available in Python through the NLTK library and can split text into sentences. When one wants to use Punkt to split texts of a certain domain it is best to train a custom Punkt model for better results. Luckily for us, Punkt simply needs raw unlabeled text to be trained on.

To train Punkt we are going to use all of the fields we extracted from each Raptarchis document whose total data equals 340MB, but this is a bit too much for Punkt to train

altogether. So we will split the 340MB worth of data into smaller chunks of files so that Punkt can train on each one separately. Each file that we create will be a certain number of lines long. After a Punkt model completes its training it will have produced three lists, which it will make use of to disambiguate sentence boundaries. So we need to decide on how many lines each file should contain. After trying many different configurations for what that line limit should be for each file, we decided that we would cap each file at 1000 lines (the entire dataset is 3 million lines long). This is done on our part for two reasons. The first reason is that we noticed that the quality of the lists that Punkt builts to decide later on where to split sentences worsened the more lines we added to each file. The second reason has to do with the time the model takes to train. Punkt models that were trained on files that were tens or hundred of thousand lines long trained for only 3 minutes, while the model we went with, which was trained on files capped at 1000 lines took half an hour. In other words, we think that by making Punkt train longer we would be able to achieve better quality support lists for Punkt. After the training is done, we simply save our Punkt model and load it when we want to split a text into sentences.

4. MODEL EVALUATION

In this chapter we are going to judge the quality of each model's sentence splitting. We will do this by using certain examples taken from the dataset, and discuss how close each SBD model will be to the optimal sentence splitting. After weighing each model's shortcomings we will choose one of them to produce an annotated version of our dataset with sentence boundaries.

4.1 Rule-based Model Evaluation

4.1.1 Example 1: Simple Sentence Splitting

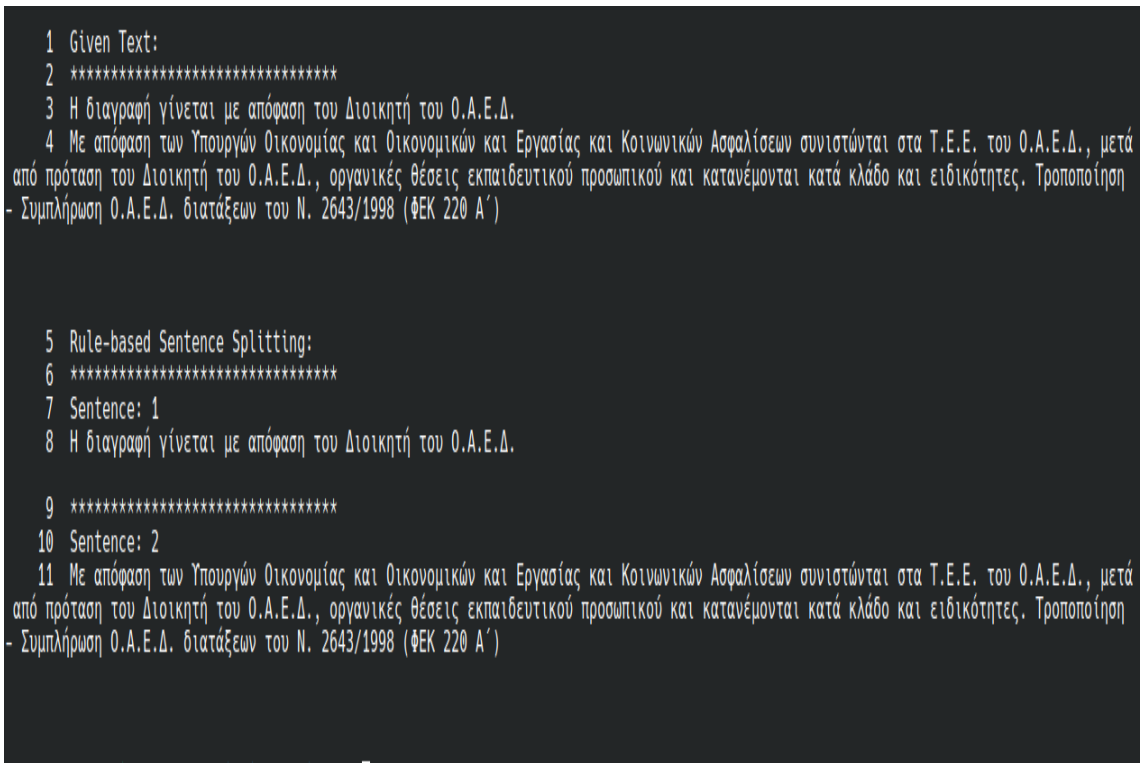

Figure 4.1: Example 1 splitted by our Rule-based Method with Numbered Lines

In this example, our rule-based method almost achieved the entirely correct outcome. The first case of ambiguity is the period that ends the line number 3. Here, the rule-based method recognizes that the word of this potential boundary point is an abbreviation of type 1 so it looks to the first word of the line number 4. Since the word 'Με' is a frequent-sentence starter, the heuristic marks the period of the last line as sentence ending.

The second case of ambiguity is the 'T.T.E.' abbreviation found in line number 4. Here, the rule-based method recognizes that the last period of the abbreviation does not end a sentence, since the word 'του' is not deemed to be a sentence starter.

The case where the method fails to mark a sentence boundary is towards the end of line 4, where we have a period followed by the word "Τροποποίηση". Since that word is not a sentence-starter based on our method the period is not marked as sentence ending, the problem here is that the period is sentence ending and "Τροποποίηση" starts the next sentence.

4.1.2 Example 2: Recognizing Enumerated lists

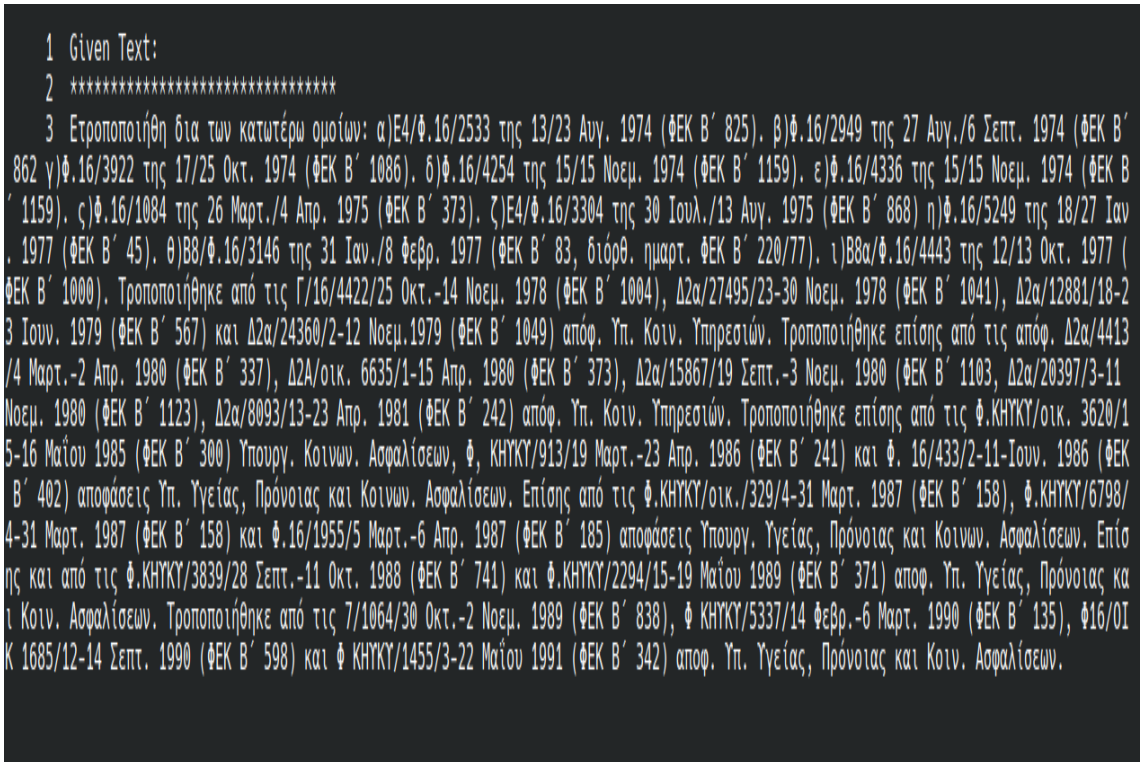

Figure 4.2: Example 2 before being splitted

In this example, our rule-based method achieves the entirely correct outcome. The tricky thing here is to recognize each member of the enumerated list. The important thing here is that the heuristic we used for recognizing enumerated lists that use right parentheses works and correctly marks each member as its own unique sentence, furthermore its able to recognize the words ‘Τροποποιήθηκε’ and ‘Επίσης’ as sentence starters. This is a great example that showcases the ability of our rule-based method to split sentences that are not separated with newlines. One last thing is that sometimes this method fails to identify some enumerated lists. In particular, when the start of each member is not included in the frequent-sentence starters, or they are not introduced using right parentheses, making our ability to recognize enumerated lists not always perfect.

4.1.3 Example 3: Recognizing Enumerated lists with periods introducing its list item

Example number 3 is one big sentence that has many enumerated lists. Here, we start noticing the limits of our rule-based approach. Our rule-based approach fails to recognize that in line number 7 we have an enumerated list introduction, simply because the right word (‘1.Το’) is not present in our frequent-sentence-starters list. In comparison, line number 10 shows the second list entry being recognized. This happens a) because there is a period and a space before it, which makes our main-heuristic identify it as a candidate sentence boundary point, and b) because (’2.Την’) is present in our frequent-starters-list. Whenever (a) or (b) happen, we miss a potential sentence boundary point. As we can see in line number 13, there are many list entries that are not being identified, because, while there is a period before introducing them, there, starting words are not considered

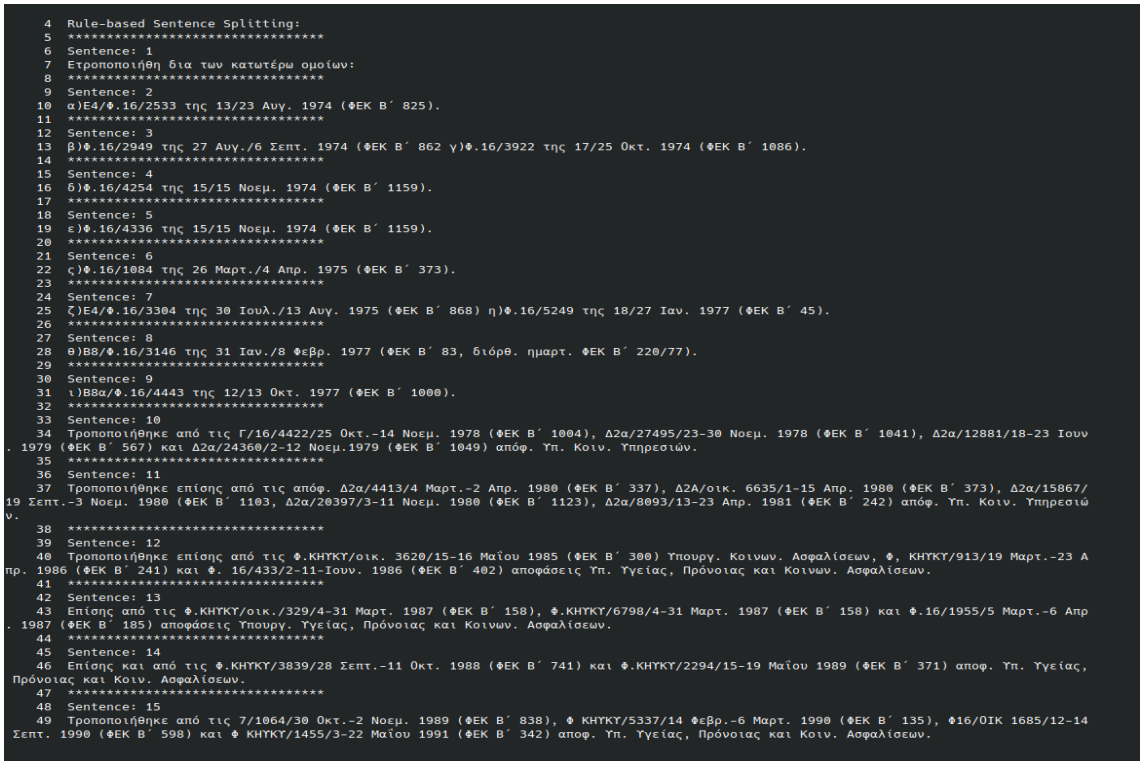

Figure 4.3: Example 2 splitted by our Rule-based Method with Numbered Lines

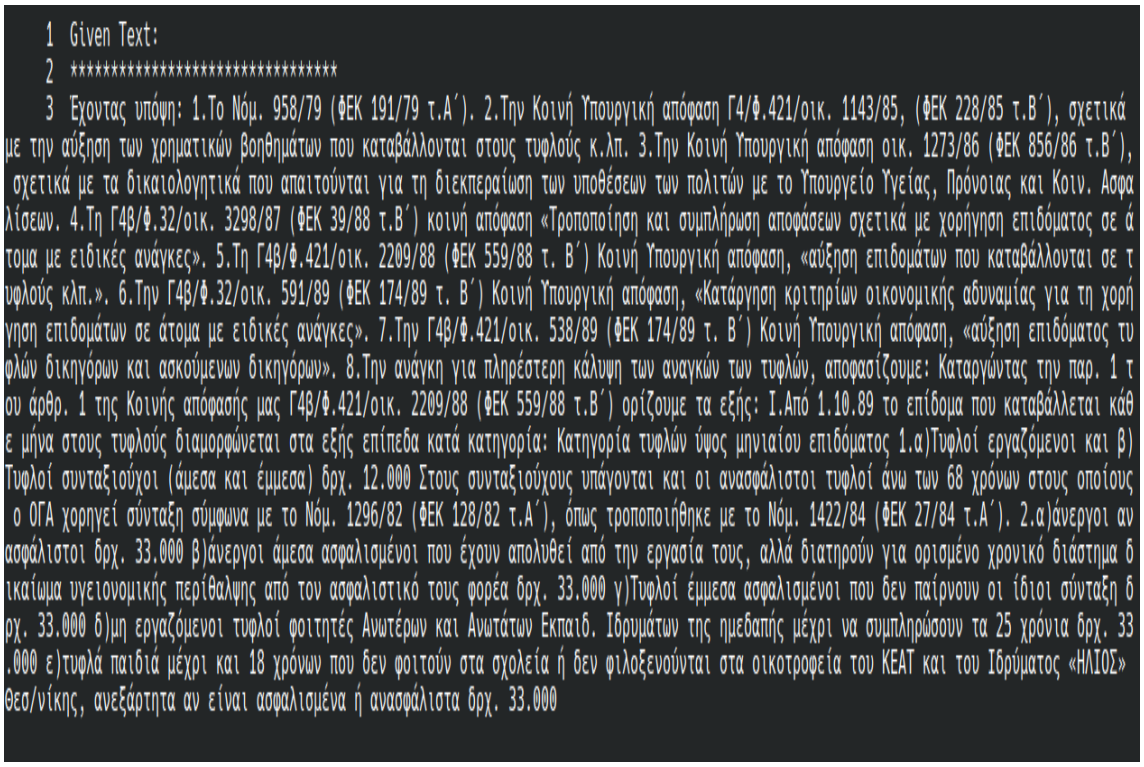

Figure 4.4: Example 3 before being splitted

```
    4  Rule-based Sentence Splitting:
    5  **********
    6  Sentence: 1
    7  Έχοντας υπόψη: 1.Το Νόμ. 958/79 (ΦΕΚ 191/79 τ.Α΄).
    8  **********
    9  Sentence: 2
   10  2.Την Κοινή Υπουργική απόφαση Γ4/Φ.421/οικ. 1143/85, (ΦΕΚ 228/85 τ.Β΄), σχετικά με την αύξηση των χρηματικών βοηθημάτων που καταβάλ
λονται στους τυφλούς κ.λπ.
   11  **********
   12  Sentence: 3
   13  3.Την Κοινή Υπουργική απόφαση οικ. 1273/86 (ΦΕΚ 856/86 τ.Β΄), σχετικά με τα δικαιολογητικά που απαιτούνται για τη διεκπεραίωση των
υποθέσεων των πολιτών με το Υπουργείο Υγείας, Πρόνοιας και Κοιν. Ασφαλίσεων. 4.Τη Γ4β/Φ.32/οικ. 3298/87 (ΦΕΚ 39/88 τ.Β΄) κοινή απόφαση «Τρο
ποποίηση και συμπλήρωση αποφάσεων σχετικά με χορήγηση επιδόματος σε άτομα με ειδικές ανάγκες». 5.Τη Γ4β/Φ.421/οικ. 2209/88 (ΦΕΚ 559/88 τ. Β
΄) Κοινή Υπουργική απόφαση, «αύξηση επιδομάτων που καταβάλλονται σε τυφλούς κλπ.». 6.Την Γ4β/Φ.32/οικ. 591/89 (ΦΕΚ 174/89 τ. Β΄) Κοινή Υπου
ργική απόφαση, «Κατάργηση κριτηρίων οικονομικής αδυναμίας για τη χορήγηση επιδομάτων σε άτομα με ειδικές ανάγκες».
   14  **********
   15  Sentence: 4
   16  7.Την Γ4β/Φ.421/οικ. 538/89 (ΦΕΚ 174/89 τ. Β΄) Κοινή Υπουργική απόφαση, «αύξηση επιδόματος τυφλών δικηγόρων και ασκούμενων δικηγόρω
ν». 8.Την ανάγκη για πληρέστερη κάλυψη των αναγκών των τυφλών, αποφασίζουμε: Καταργώντας την παρ. 1 του άρθρ. 1 της Κοινής απόφασής μας Γ4β
/Φ.421/οικ. 2209/88 (ΦΕΚ 559/88 τ.Β΄) ορίζουμε τα εξής: Ι.Από 1.10.89 το επίδομα που καταβάλλεται κάθε μήνα στους τυφλούς διαμορφώνεται στα
 εξής επίπεδα κατά κατηγορία:
   17  **********
   18  Sentence: 5
   19  Κατηγορία τυφλών ύψος μηνιαίου επιδόματος 1.α)Τυφλοί εργαζόμενοι και β)Τυφλοί συνταξιούχοι (άμεσα και έμμεσα) δρχ. 12.000 Στους συν
ταξιούχους υπάγονται και οι ανασφάλιστοι τυφλοί άνω των 68 χρόνων στους οποίους ο ΟΓΑ χορηγεί σύνταξη σύμφωνα με το Νόμ. 1296/82 (ΦΕΚ 128/8
2 τ.Α΄), όπως τροποποιήθηκε με το Νόμ. 1422/84 (ΦΕΚ 27/84 τ.Α΄). 2.α)άνεργοι ανασφάλιστοι δρχ. 33.000 β)άνεργοι άμεσα ασφαλισμένοι που έχου
ν απολυθεί από την εργασία τους, αλλά διατηρούν για ορισμένο χρονικό διάστημα δικαίωμα υγειονομικής περίθαλψης από τον ασφαλιστικό τους φορ
έα δρχ. 33.000 γ)Τυφλοί έμμεσα ασφαλισμένοι που δεν παίρνουν οι ίδιοι σύνταξη δρχ. 33.000 δ)μη εργαζόμενοι τυφλοί φοιτητές Ανωτέρων και Ανω
τάτων Εκπαιδ. Ιδρυμάτων της ημεδαπής μέχρι να συμπληρώσουν τα 25 χρόνια δρχ. 33.000 ε)τυφλά παιδιά μέχρι και 18 χρόνων που δεν φοιτούν στα
σχολεία ή δεν φιλοξενούνται στα οικοτροφεία του ΚΕΑΤ και του Ιδρύματος «ΗΛΙΟΣ» Θεσ/νίκης, ανεξάρτητα αν είναι ασφαλισμένα ή ανασφάλιστα δρχ
. 33.000
```

Figure 4.5: Example 3 splitted by our Rule-based Method with Numbered Lines

sentence starters. The only one being recognized is the one in line 16. Finally, the enumerated list contained in line number 19 is not being recognized because the list entries are separated by space characters, but there is no heuristic method that could pick up on this, because of the lack of a candidate sentence boundary points.

4.1.4 Performance

We consider the sentence splitting of the rule-based method to be above average. For starters, our rule-based method performs better than sentence segmenting tools that have not been developed for our specific domain, and simply require more effort to get them to work well for a specific domain and language, like PySBD [14]. Furthermore, there are some cases where, the combination of our support resources along with our heuristics fail to recognize some sentence boundary points, but we still consider our method well performing on the Raptarchis dataset because all support lists and heuristics are built around words that are most commonly seen in the dataset, whether those are the most common sentence-starters or the most common ways abbreviations appear in the text. In other words, we can miss sentence boundaries but we do not miss the majority of them.

4.2 Punkt Model Evaluation

4.2.1 Example 1: Simple Sentence Splitting

In this example, our Punkt Model gets the disambiguation of the first candidate sentence boundary point in line 8 incorrect. The reason why this happens is the fact that the abbreviation ‘Ο.Α.Ε.Δ.’ is present in Punkt’s abbreviation list, therefore, Punkt will look to the

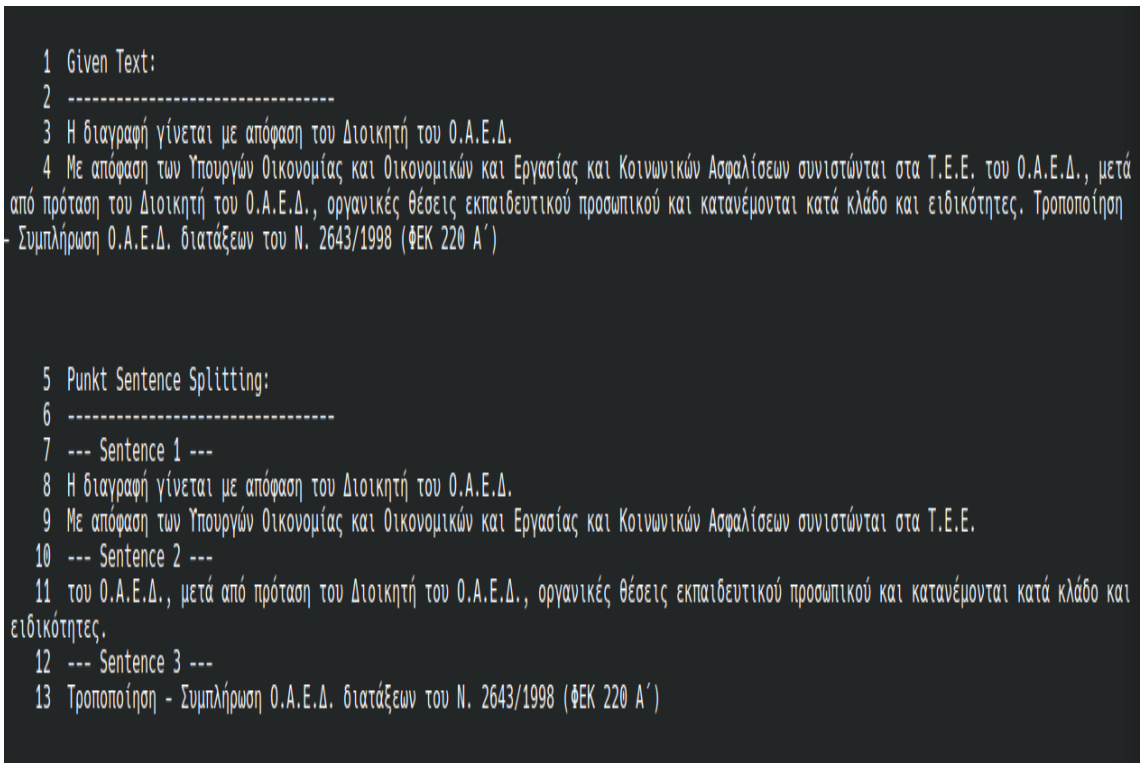

Figure 4.6: Example 1 splitted by our Punkt Model with Numbered Lines

right word ‘Με’ to decide if it will the last period of ‘Ο.Α.Ε.Δ.’ as sentence ending. Since ‘Με’ is not included in that list, Punkt decides not to mark a sentence boundary here. The second mistake is on line 9 and is the fact that the last period of the abbreviation ‘Τ.Ε.Ε.’ is recognized as sentence ending, this happens simply because the Punk model we trained does not include this specific abbreviation to its abbreviations list. Punkt is able to recognize the last sentence in the text correctly which is something our Rule-based model failed to do.

4.2.2 Example 2: Recognizing Enumerated lists

In this example, our Punkt Model fails to recognize that in line 7 there is an enumerated list introduction, but succeeds in recognizing all the other list entries of the enumerated as their own sentences, but we start noticing some incorrectly tagged boundaries. In particular, in lines 15 and 23 we notice that our Punkt Model did not recognize some of the most common type 2 abbreviations, which are month names and again, the same problem seems to be repeated on subsequent lines like 49,51, and others.

4.2.3 Example 3: Recognizing Enumerated lists with periods introducing its list item

Again we have the problem that some type 2 abbreviations that have not been recognized end sentences, but we see that Punkt is able to recognize the start of every list entry of the enumerated list in the example not including the first one in line 7. Another thing is that in lines 53, 55, 57, 61, Punkt fails to recognize that we have enumerated list entries. The only reason that the enumerated list is separated is because each abbreviation that is part of the previous list entry is classified as a sentence boundary, which is incorrect.

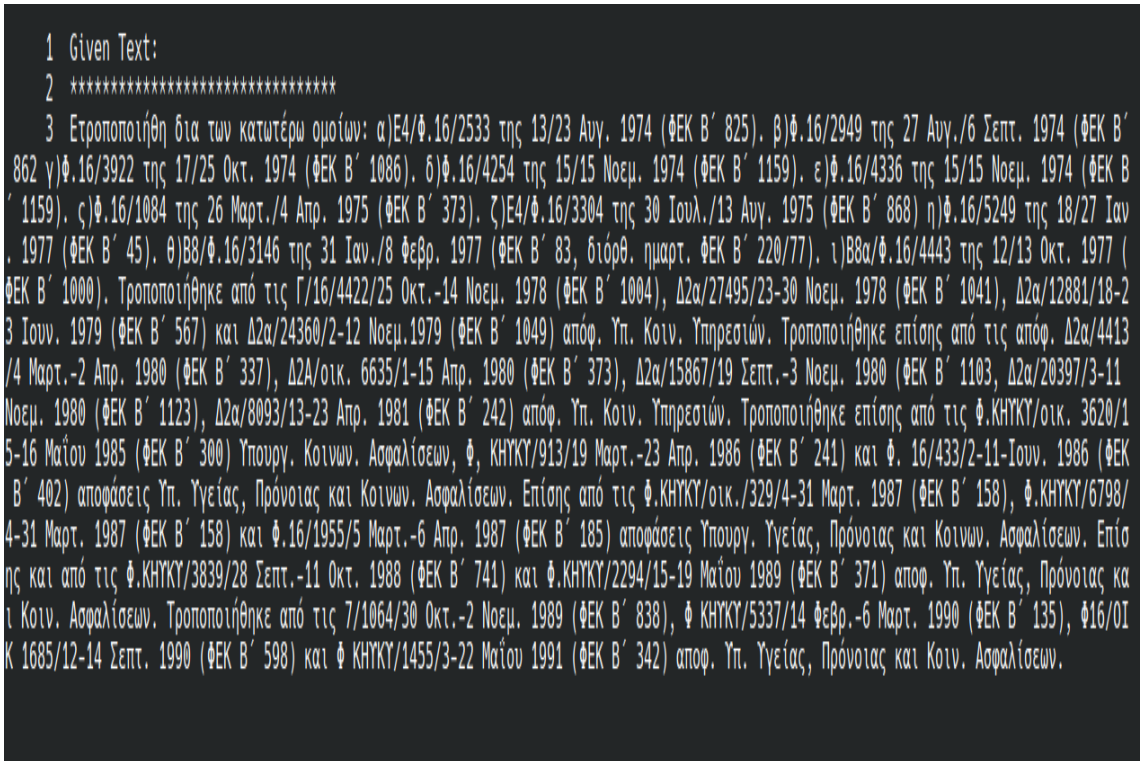

Figure 4.7: Example 2 before being splitted

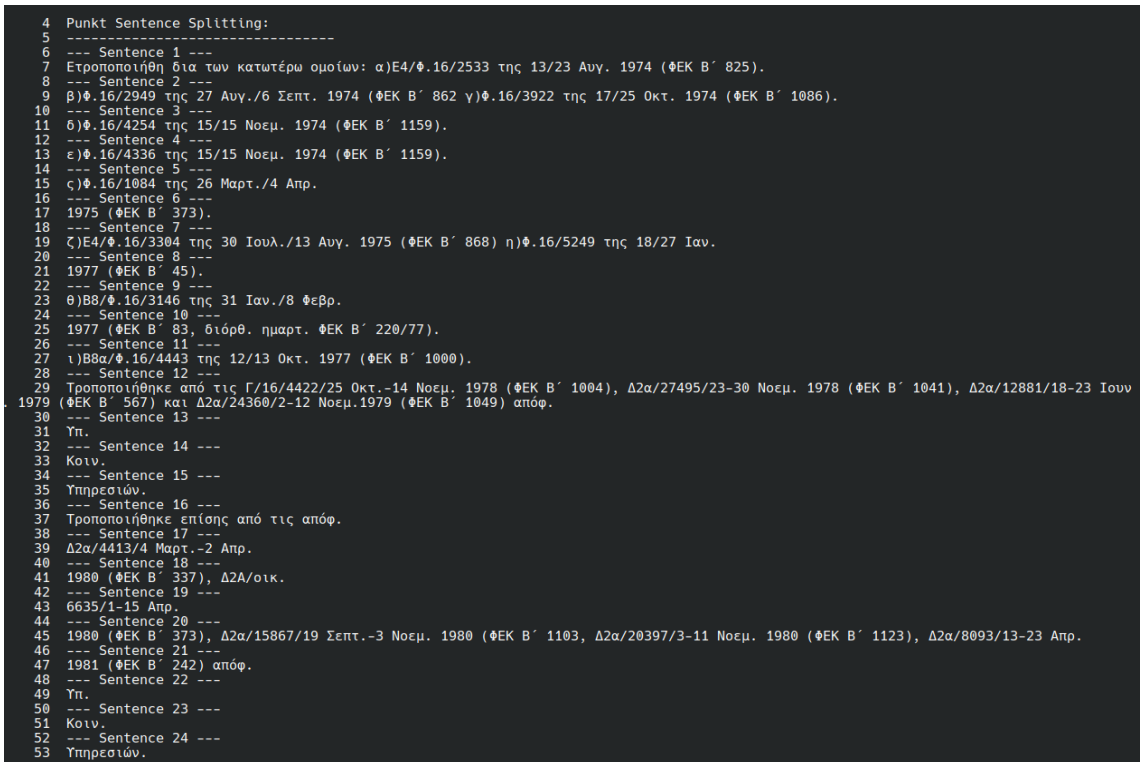

Figure 4.8: Example 2 splitted by our Punkt model with Numbered Lines

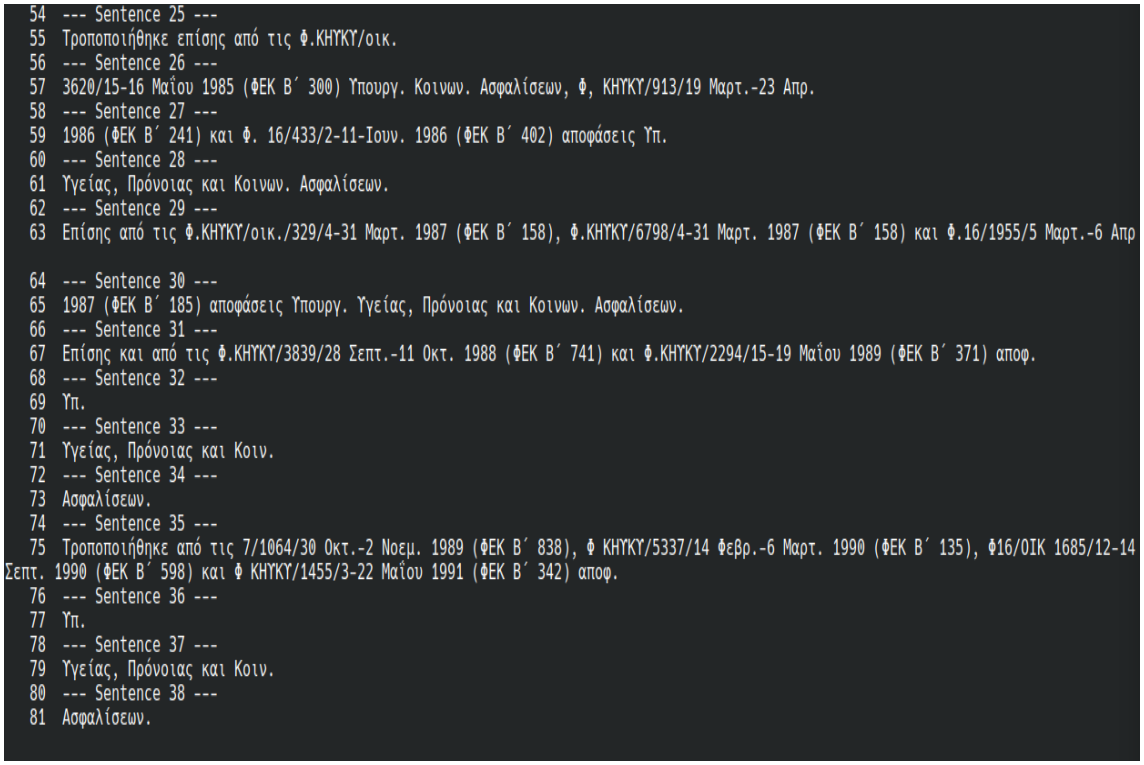

Figure 4.9: Example 2 splitted by our Punkt model with Numbered Lines

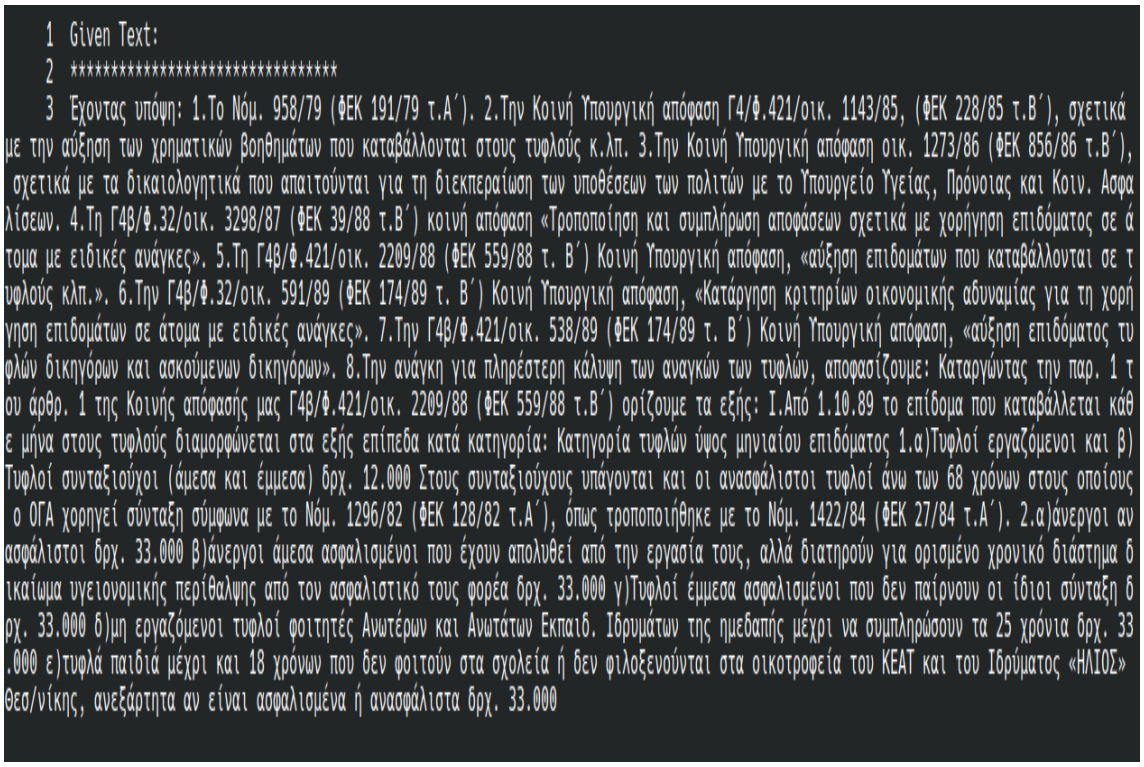

Figure 4.10: Example 3 before being splitted

```
Punkt Sentence Splitting:
---------------------------------
--- Sentence 1 ---
Έχοντας υπόψη: 1.Το Νόμ.
--- Sentence 2 ---
958/79 (ΦΕΚ 191/79 τ.Α΄).
--- Sentence 3 ---
2.Την Κοινή Υπουργική απόφαση Γ4/Φ.421/οικ.
--- Sentence 4 ---
1143/85, (ΦΕΚ 228/85 τ.Β΄), σχετικά με την αύξηση των χρηματικών βοηθημάτων που καταβάλλονται στους τυφλούς κ.λπ.
--- Sentence 5 ---
3.Την Κοινή Υπουργική απόφαση οικ.
--- Sentence 6 ---
1273/86 (ΦΕΚ 856/86 τ.Β΄), σχετικά με τα δικαιολογητικά που απαιτούνται για τη διεκπεραίωση των υποθέσεων των πολιτών με το Υπουργείο Υγείας, Πρόνοιας και Κοιν.
--- Sentence 7 ---
Ασφαλίσεων.
--- Sentence 8 ---
4.Τη Γ4β/Φ.32/οικ.
--- Sentence 9 ---
3298/87 (ΦΕΚ 39/88 τ.Β΄) κοινή απόφαση «Τροποποίηση και συμπλήρωση αποφάσεων σχετικά με χορήγηση επιδόματος σε άτομα με ειδικές ανάγκες».
--- Sentence 10 ---
5.Τη Γ4β/Φ.421/οικ.
--- Sentence 11 ---
2209/88 (ΦΕΚ 559/88 τ. Β΄) Κοινή Υπουργική απόφαση, «αύξηση επιδομάτων που καταβάλλονται σε τυφλούς κλπ.».
--- Sentence 12 ---
6.Την Γ4β/Φ.32/οικ.
--- Sentence 13 ---
591/89 (ΦΕΚ 174/89 τ. Β΄) Κοινή Υπουργική απόφαση, «Κατάργηση κριτηρίων οικονομικής αδυναμίας για τη χορήγηση επιδομάτων σε άτομα με ειδικές ανάγκες».
--- Sentence 14 ---
7.Την Γ4β/Φ.421/οικ.
--- Sentence 15 ---
538/89 (ΦΕΚ 174/89 τ. Β΄) Κοινή Υπουργική απόφαση, «αύξηση επιδόματος τυφλών δικηγόρων και ασκούμενων δικηγόρων».
--- Sentence 16 ---
8.Την ανάγκη για πληρέστερη κάλυψη των αναγκών των τυφλών, αποφασίζουμε: Καταργώντας την παρ.
--- Sentence 17 ---
1 του άρθρ.
--- Sentence 18 ---
1 της Κοινής απόφασής μας Γ4β/Φ.421/οικ.
--- Sentence 19 ---
2209/88 (ΦΕΚ 559/88 τ.Β΄) ορίζουμε τα εξής: Ι.Από 1.10.89 το επίδομα που καταβάλλεται κάθε μήνα στους τυφλούς διαμορφώνεται στα εξής επίπεδα κατά κατηγορία: Κατηγορία τυφλών ύψος μηνιαίου επιδόματος 1.α)Τυφλοί εργαζόμενοι και β)Τυφλοί συνταξιούχοι (άμεσα και έμμεσα) δρχ
.
```

Figure 4.11: Example 3 splitted by our Punkt Model with Numbered Lines

```
--- Sentence 20 ---
12.000 Στους συνταξιούχους υπάγονται και οι ανασφάλιστοι τυφλοί άνω των 68 χρόνων στους οποίους ο ΟΓΑ χορηγεί σύνταξη σύμφωνα με το Νόμ.
--- Sentence 21 ---
1296/82 (ΦΕΚ 128/82 τ.Α΄), όπως τροποποιήθηκε με το Νόμ.
--- Sentence 22 ---
1422/84 (ΦΕΚ 27/84 τ.Α΄).
--- Sentence 23 ---
2.α)άνεργοι ανασφάλιστοι δρχ.
--- Sentence 24 ---
33.000 β)άνεργοι άμεσα ασφαλισμένοι που έχουν απολυθεί από την εργασία τους, αλλά διατηρούν για ορισμένο χρονικό διάστημα δικαίωμα υγειονομικής περίθαλψης από τον ασφαλιστικό τους φορέα δρχ.
--- Sentence 25 ---
33.000 γ)Τυφλοί έμμεσα ασφαλισμένοι που δεν παίρνουν οι ίδιοι σύνταξη δρχ.
--- Sentence 26 ---
33.000 δ)μη εργαζόμενοι τυφλοί φοιτητές Ανωτέρων και Ανωτάτων Εκπαιδ.
--- Sentence 27 ---
Ιδρυμάτων της ημεδαπής μέχρι να συμπληρώσουν τα 25 χρόνια δρχ.
--- Sentence 28 ---
33.000 ε)τυφλά παιδιά μέχρι και 18 χρόνων που δεν φοιτούν στα σχολεία ή δεν φιλοξενούνται στα οικοτροφεία του ΚΕΑΤ και του Ιδρύματος «ΗΛΙΟΣ» Θεσ/νίκης, ανεξάρτητα αν είναι ασφαλισμένα ή ανασφάλιστα δρχ.
--- Sentence 29 ---
33.000
```

Figure 4.12: Example 3 splitted by our Punkt Model with Numbered Lines

4.2.4 Analyzing Punkt Results

After Punkt is trained, it has created 3 lists, a list of abbreviations, a list of sentence-starters and a list of collocations. Every time the Punkt model is used it will make use of these 3 lists to decide on sentence boundaries almost like our rule-based method.

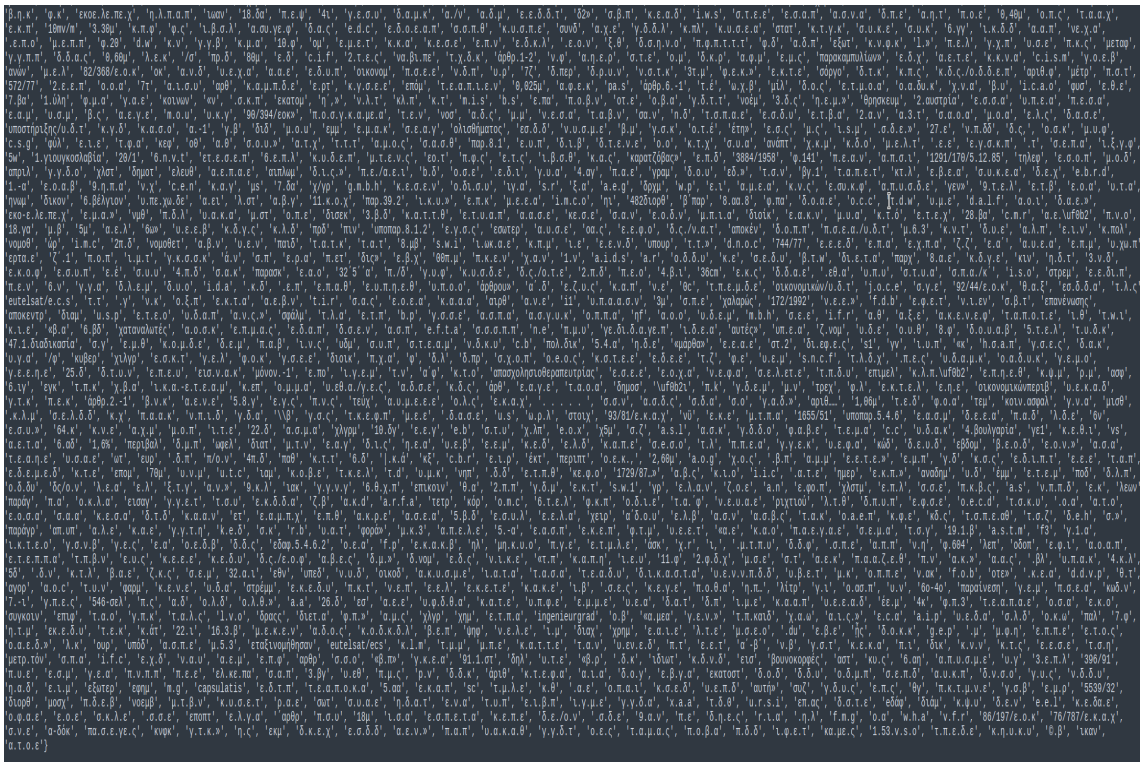

Figure 4.13: Part of the abbreviation list of our Punkt Model

We are going to discuss the advantages and disadvantages of our Punkt Model, based on how it performed on splitting the examples provided, what limits the system from getting the best split on the sentences it was given, and how it compares with the rule-based method.

Advantages:

1. One advantage of Punkt is that it can find abbreviations in the text that are not properly separated. For example: 'γ.ε.ς.,γ.ε.ν': here we have 2 abbreviations that are comma separated and that Punkt is able to identify.

2. Another advantage is that Punk is in some way able to handle typos better than our rule-based method. Punkt includes in its list of abbreviations a word like '.M.K' which is a word that is difficult to classify as an abbreviation because it begins with a period, but this word usually appears in the text with a 'Υ .M.K.', so Punkt is able to recognize a collocation between the 2 and thus classifies it as an abbreviation.

Disadvantages:

1. One disadvantage of Punkt abbreviations list is that its list includes a huge number of abbreviations of type 2 that are sometimes found 2 or 3 times in the text, and fails to include some important abbreviations like month names, or even type 2 abbreviation that are found often, but for some reason are not included in the list.

2. Second is the fact that Punkt recognizes some abbreviations that simply are not abbreviations of any type, but because Punkt sees a collocation of these abbreviation with other characters, it includes them in its list, Example: ‘π.υ.ς.»’, ‘.σ.ε.ς.3’.

All in all we believe that while Punkt does a good job with sentence splitting, it still has some flaws. A lot of things that work in its favor seem to also work against it. The recognition of certain unique kinds of abbreviations like those in the advantages also means that some bad ones like those in the disadvantages are included. Moreover, it is troubling that Punkt was not able to include some common abbreviations like month names. These are so regularly seen in the text and Punkt is not able to identify all of them.

Maybe the fact that our Punkt model has some noticeable flaws has to do with the fact that we did not train it on enough data. We did try to squeeze as much from our data as we could by trying many ways of splitting our dataset in files.

Finally, the quality of Punkt’s abbreviation list could have been improved manually by us. Punkt allows users to add a custom list of abbreviations to Punkt’s abbreviations. The reason we did not do that is because the biggest advantage of Punkt is that it works on raw unlabeled data, meaning that the user should not try to find abbreviations manually. The only reason we have such a set is because of our rule-based method.

4.3 Annotating Raptarchis with our Sentence Boundaries

Both of our SBD models have benefits and drawbacks. We could say that Punkt deals with edge cases in the dataset better with its ability to recognize some abbreviations that are not so distinct for a rule-based-method, but due to its inability to recognize common abbreviations its performance falls off a little. Furthermore, while Punkt is a great model that is easy to train and easy to use, its performance is somewhat bounded by the data we have available to train it. On the other hand, we have a rule-based system that was a bit more difficult to develop, but due to it being highly specialized for our dataset, it seems to get better results most of the time, as seen in the examples . In the end, we are choosing our rule-based model to annotate Raptarchis with sentences. The ability to use basically our support lists in conjunction with our heuristics gives us a more stable SBD system. The final part of this thesis is to split every document into sentences, and then create a ‘.json’ file for each one.

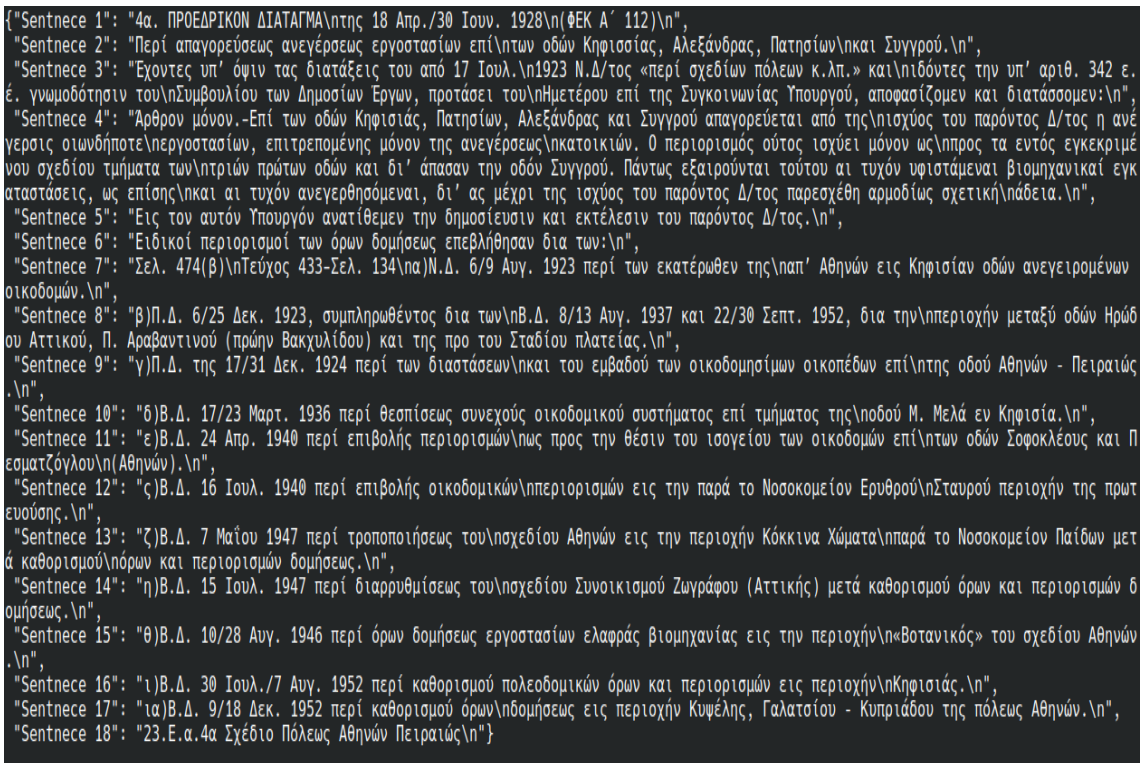

Figure 4.14: Example json file

5. CONCLUSIONS AND FUTURE WORK

At the beginning of this thesis, we examined all the approaches that have been used for SBD over the years. We also explored how legal documents are annotated with sentences in the U.S. We continued by applying the SBD methods that were best for the dataset Raptarchis. We explored all the details about this dataset, the structure of it, and its distinct points. After that, continued with the creation of two SBD models, one was a handcrafted rule-based system and the other was one based on the Punkt architecture. We evaluated both of the models and their shortcomings. Finally, we choose the Rule-based Method to annotate Raptarchis with sentence boundaries.

For the future, this annotated dataset can provide a solid basis for any further SBD work on the Raptarchis dataset.

ABBREVIATIONS - ACRONYMS

NLP	Natural Language Processing
SBD	Sentence Boundary Detection
WSJ	Wall street Journal
DCA	Document Centered Approach

BIBLIOGRAPHY

[1] Punctuation in different languages different punctuation. https://toppandigital.com/translation-blog/punctuation-in-different-languages/.

[2] Savelka github. https://github.com/jsavelka/sbd_adjudicatory_dec.

[3] Sentence boundary detection in adjudicatory decisions in the united states. https://scholarlycommons.law.hofstra.edu/cgi/viewcontent.cgi?article=2325&context=faculty_scholarship.

[4] Steven Bird. NLTK: the natural language toolkit. In Nicoletta Calzolari, Claire Cardie, and Pierre Isabelle, editors, *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*. The Association for Computer Linguistics, 2006.

[5] W. N. Francis and H. Kucera. Brown corpus manual. Technical report, Department of Linguistics, Brown University, Providence, Rhode Island, US, 1979.

[6] Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. GENIA corpus - a semantically annotated corpus for bio-textmining. In *Proceedings of the Eleventh International Conference on Intelligent Systems for Molecular Biology, June 29 - July 3, 2003, Brisbane, Australia*, pages 180–182, 2003.

[7] Tibor Kiss and Jan Strunk. Unsupervised multilingual sentence boundary detection. *Comput. Linguistics*, 32(4):485–525, 2006.

[8] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Comput. Linguistics*, 19(2):313–330, 1993.

[9] Andrei Mikheev. Periods, capitalized words, etc. *Comput. Linguistics*, 28(3):289–318, 2002.

[10] David D. Palmer. SATZ - an adaptive sentence segmentation system. *CoRR*, cmp-lg/9503019, 1995.

[11] Christos Papaloukas, Ilias Chalkidis, Konstantinos Athinaios, Despina-Athanasia Pantazi, and Manolis Koubarakis. Multi-granular legal topic classification on greek legislation. *CoRR*, abs/2109.15298, 2021.

[12] Jonathon Read, Rebecca Dridan, Stephan Oepen, and Lars Jørgen Solberg. Sentence boundary detection: A long solved problem? In Martin Kay and Christian Boitet, editors, *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Posters, 8-15 December 2012, Mumbai, India*, pages 985–994. Indian Institute of Technology Bombay, 2012.

[13] Michael D. Riley. Some applications of tree-based modelling to speech and language. In *Speech and Natural Language: Proceedings of a Workshop Held at Cape Cod, Massachusetts, USA, HLT 1989, October 15-18, 1989*. ACL, 1989.

[14] Nipun Sadvilkar and Mark Neumann. Pysbd: Pragmatic sentence boundary disambiguation. *CoRR*, abs/2010.09657, 2020.

[15] Jörg Tiedemann. Parallel data, tools and interfaces in OPUS. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Ugur Dogan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012*, pages 2214–2218. European Language Resources Association (ELRA), 2012.