DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Data Engineering and Analytics

Automatic English Text Simplification for Statutes

Akshaya Muralidharan

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Data Engineering and Analytics

Automatic English Text Simplification for Statutes

Automatische Textvereinfachung für englische Gesetze

Author:	Akshaya Muralidharan
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Stefanie Urchs
Submission Date:	October 15th, 2021

I confirm that this master's thesis in data engineering and analytics is my own work and I have documented all sources and material used.

Munich, October 15th, 2021 Akshaya Muralidharan

Abstract

Every human being at some point of their life must deal with the domestic or international legal system and looks for information on laws and regulations. However, due to the complex language in which statutes are presented, they fail to obtain the knowledge they seek. In this thesis, we create a corpus of simple legal text and do a preliminary examination of what it means to simplify legal information.

We create a corpus of simple legal text and its corresponding laws in standard form. We then compare the properties of the two kinds of legal texts in terms of text complexity scores and grammatical structure. We show that properties of simple generic text do not always apply to simple legal text. Further, we explore current state-of-the-art text simplification models and examine how they transform or simplify legal text. We analyse the simplified statutes produced by the models and determine what type of transformations suit the task of legal text simplification. This research motivates the necessity for specialised legal text simplification models and provides guidelines for defining the objectives of such models.

Kurzfassung

Jeder Mensch hat irgendwann in seinem Leben mit dem nationalen oder internationalen Rechtssystem zu tun und sucht nach Informationen über Gesetze und Vorschriften. Aufgrund der komplexen Sprache, in der Gesetze präsentiert werden, scheitert sie/er jedoch daran, das gesuchte Wissen zu erlangen. In dieser Arbeit erstellen wir ein Korpus einfacher Rechtstexte und führen eine Voruntersuchung darüber durch, was es bedeutet, rechtliche Informationen zu vereinfachen.

Wir erstellen einen Korpus mit einfachen Rechtstexten und den entsprechenden Gesetzen in Standardform. Anschließend vergleichen wir die Eigenschaften der beiden Arten von Rechtstexten im Hinblick auf Textkomplexitätsmetriken und grammatikalische Struktur. Wir zeigen, dass die Eigenschaften einfacher allgemeiner Texte nicht immer auf einfache Rechtstexte zutreffen. Darüber hinaus untersuchen wir den aktuellen Stand der Technik bei Textvereinfachungsmodellen und untersuchen, wie sie Rechtstexte transformieren oder vereinfachen. Wir analysieren die von den Modellen erzeugten vereinfachten Gesetze und bestimmen, welche Art von Transformationen für die Aufgabe der Vereinfachung von Rechtstexten geeignet ist. Diese Forschung motiviert die Notwendigkeit von spezialisierten Modellen zur Vereinfachung von Rechtstexten und liefert Leitlinien für die Definition der Ziele solcher Modelle.

Contents

1. Introduction

Text simplification is a set of transformations one can perform on a body of text to make it easier to understand. It can include a variety of changes to the text, such as changing the words of the text, changing its grammatical structure, changing the format of the text, or providing additional information such that it becomes easier to read and understand while the original meaning remains unchanged. The end use-case of the simplification process can vary greatly. However, the purpose of text simplification is to make textual information more accessible. Some transformations, such as triple extraction, where one extracts triples of (subject, predicate, object) from the text, make information more accessible for automation. Other text simplification transformations intend to make information easier to access for human readers. It involves many varieties of linguistic transformations such as lexical simplification, where we transform complex words to simple words, paraphrasing, where the structure of the text is changed, and tense and voice simplification, among others. Another simplification transformation is to split sentences into multiple smaller easier-to-read sentences. Summarization is another transformation where the goal is similar - to make the text easier to understand. However, the summarization task tolerates loss of information whereas, in the simplification task, the goal is to have identical information as the source text. All these tasks aim to make the text more understandable to human readers.

Text simplification is especially crucial in the legal domain, where government regulations are often written in a hard-to-understand language. It has been shown that using simple language makes justice more accessible [1]. Several services, such as Capito[1], offer paid services to convert legal text to a simplified form, but these are time consuming and expensive options that are not accessible to the common man. There aren't any publicly available AI-assisted services to do this in an automated or semi-automated manner. To build or study such systems, it is crucial to have a dataset of aligned complex and simple legal texts. While such corpora do not exist at the moment, there are many legal articles online which contain simplified language and could be leveraged to build an aligned corpus using various text alignment techniques. Once we have an aligned corpus, we may use this to study the text simplification task for the legal domain. It is essential to understand what kind of transformations are suitable for making legal texts accessible for human readers. For example, summarization may be unsuitable for legal text simplification since readers receive less information from the summarized text than from the source text. It could be risky for readers to receive incomplete legal information. Similarly, lexical simplification may also result in change of information since certain legal terms hold specific meanings that may change when substituted with simpler words. Analysing this corpus helps us understand

[1] `https://www.capito.eu/` accessed 08-10-2021.

which transformations are suitable for simplifying legal text.

1.1. Problem Description

In order to create systems that automatically simplify legal text, we first need to identify the transformations that turn standard legal texts into easily readable legal text. There are many linguistic transformations F_{type} that could be applied in order to simplify text.

Lexical simplification is one such linguistic transformation. For example,

$$F_{lexical}(\text{“}Is\ this\ adequate?\text{”}) = \text{“}Is\ this\ enough?\text{”}$$

Here, complex words are replaced with simpler synonyms without changing the structure of the sentence.

Some other transformation functions are: voice change, where passive voice sentences are converted to active voice; tense change, where all tenses are converted to simple present or simple past; paraphrasing, where the structure of the sentence is changed; and sentence splitting, where long compound sentences are broken down into constituents with one thought per sentence. These are only a few of the many innumerable transformations that could be performed to simplify text.

$$F_{voice}(\text{“}It\ is\ being\ handled\ by\ us.\text{”}) = \text{“}We\ are\ handling\ it.\text{”}$$

$$F_{tense}(\text{“}We\ have\ been\ looking\ into\ it.\text{”}) = \text{“}We\ looked\ into\ it.\text{”}$$

$$F_{paraphrase}(\text{“}Entry\ is\ restricted.\text{”}) = \text{“}Not\ allowed\ to\ enter.\text{”}$$

$$F_{splitting}(\text{“}While\ time\ is\ finite,\ we\ must\ not\ treat\ it\ so.\text{”}) = \text{“}Time\ is\ finite.\ But\ we\ must\ not\ treat\ it\ so.\text{”}$$

If a human were to simplify a source sentence T, they would apply an array of such transformations F_1, F_2,...,F_n to T to obtain a simplified sentence $T_{simplified}$.

$$T_{simplified} = F_n(F_{n-1}...(F_2(F_1(T)))) \tag{1.1}$$

where each F_i would represent one linguistic transformation. If we knew which linguistic transformations were suitable for legal texts, we could develop an automated system that performs each of this transformations separately and apply them. However, this is not a preferred method as it has many disadvantages such as compounded errors and necessity of multiple datasets (one for each of the transformation tasks).

Most simplification systems today are trained using deep neural networks, and the end result is obtained by performing various transformations on vectors that represent the source text. Without going too deep into the workings of a neural network, a transformation performed by typical multi-layer neural network can by represented by the following:

$$T'_{simplified} = F'_{decoding}(F'_{m-1}...(F'_2(F'_{encoding}(T)))) \tag{1.2}$$

here, $F'_{encoding}$ is a transformation that converts the source text to a numerical vector representation, F'_2,...,F'_{m-1} are various linear and non-linear operations and $F'_{decoding}$ is a transformation that maps the resulting numerical vector to text.

The various transformation functions F'_2,...,F'_{n-1} are derived by finding a set of transformations that produce simplified texts that are closest to an example dataset. To obtain a good performance in these types of models, a large amount of data is required. For example, the current state-of-the-art text simplification model "MUSS: Multilingual Unsupervised Sentence Simplification by Mining Paraphrases" [2] uses a dataset of more than 1 million samples to train their model. Given that availability of simple legal data is sparse, the feasibility of generating a dataset of this size for developing legal text simplification models using this method is questionable.

However, there exists another method of training these models known as reinforcement learning. These models depend, not on an existing dataset, but rather are continuously updated based on the signals received on the output of the intermediate model. For example, in the reinforcement learning based simplification model [3] shown in Figure 1.1 the model receives a reward based on three scores, namely simplicity, relevance and grammar, on the produced output. Based on these scores, the REINFORCE algorithm decides how to modify the model in order to receive a higher reward. To develop legal text simplification models of this type, we must reward the model when it produces simple legal text. To be able to automatically detect when the model produces simple legal, we must understand what the features of good simple legal text are and learn to detect these features.

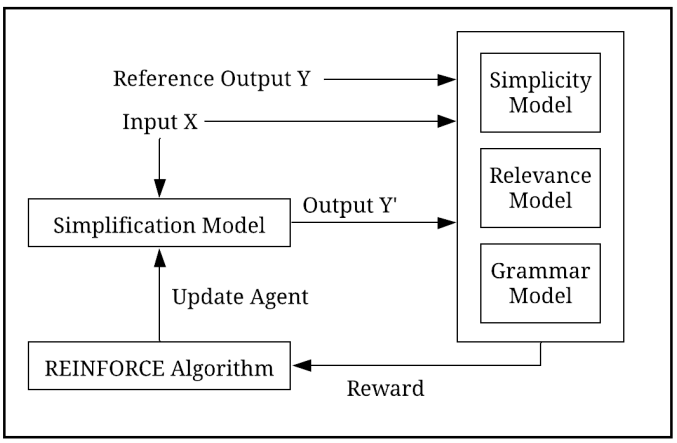

Figure 1.1.: Illustration of the DRESS Model by Zhang and Lapata. Here, Y is the reference simplified version of the input X and Y' is the output produced by the simplification model. X, Y and Y' serve as input to compute reward value based on simplicity, relevance and grammar. [3]

Even though we may not explicitly require a large dataset to train these models, we require a dataset to study the features of simple legal text as well as evaluate the output produced by such a model.

Hence, the objective of the thesis is to find a good model dataset of simple legal text and understand the features that represent it and distinguish it from standard laws. Moreover, we also need to understand how simple legal texts are different from the simplified legal texts produced by the current text simplification systems, so that we can understand how to adapt them to work for the legal domain.

1.2. Research questions

Breaking down this problem statement into more atomic components leads us to three research questions.

> **Research question 1:** How do we develop a dataset that represents accessible and simple legal information?

Answering this question involves understanding what a good source of accessible and simple legal information is and finding this source, acquiring this data and preparing it for study and technical use.

> **Research question 2:** What are the characteristics of accessible or simplified legal text?

In order to characterize simple legal text, we look at various methods of characterizing complexity of text and extend it to the legal domain and the collected data.

> **Research question 3:** How well do existing text simplification techniques perform on legal text?

Automatic text simplification is a problem that is being tackled by researchers for various purposes from as early as 1999 when Carroll et al. [4] defined the various parts and sub-processes that a potential automatic text simplification system must have. Naturally, this implies that there exist state-of-the-art text simplification systems that perform this task to varying degrees of success. However, currently, there exists no information regarding the applicability of the generic text simplification systems to legal texts. Hence, an essential task here is to understand how similar or different the automatic text simplification task is compared to the task of legal text simplification. This understanding can help guide the next wave of research towards either adapting state-of-the-art simplification systems for the legal domain or towards developing new datasets, evaluation criteria, and legal text simplification systems tailored exclusively for this domain.

1.3. Thesis Outline

The rest of this thesis is organized as follows:

Chapter 2 provides a brief look into existing text simplification services and guidelines. It then explores potential data sources of simple legal text and different text alignment methods commonly used in literature to create monolingual aligned text datasets. In this chapter, we also explore some methods of measuring and characterizing the difficulty of texts. This chapter also provides a look into the state-of-the-art in automatic text simplification and existing simplification corpora.

Chapter 3 explains the shortcomings of existing datasets for the simplification task and motivates the necessity of a domain specific dataset. It explores the dataset creation pipeline in detail and describes the tools used to create the simple and standard legal datasets. It then goes on to explain how the simple and standard legal datasets are aligned. Finally, the chapter presents a discussion on the limitations of the created dataset and how it could potentially be improved.

Chapter 4 explores and motivates the metrics used to measure the readability of the corpus. The chapter also introduces other features, not previously used in literature, which are used in this thesis to characterize complexity of the legal text. It then presents the results and provides the interpretation of these characteristics. Finally, the chapter presents a discussion of these results and their implication for the task of legal text simplification.

Chapter 5 takes a look at the existing automatic text simplification models and how we use them to generate simplifications of legal text. It compares the simplified text produced by the models quantitatively and qualitatively and attempts to provide an interpretation of the complexity scores of the simplified texts. Finally, it presents a discussion on what these results imply for the task of legal text simplification.

Chapter 6 concludes the thesis and presents potential directions the next step of research could take for solving the task of text simplification for legal texts.

2. Related Work

In this chapter, we look at some guidelines on how to create simple text from various sources. We then explore the currently available services and avenues that provide access to simpler legal information. We look at paid services as well as voluntary services that help translate complex legal text into simpler text. We then explore sources of simple legal texts in the form of articles or essays. Following this, we explore various text alignment methods used in literature to create monolingual text corpora. We then list various existing text simplification corpora, where they are collected from, how large they are, and how they're collected. In the later sections, we look at various ways of measuring the complexity of text. Lastly, we take a look at different classes of text simplification techniques and state-of-the-art in these classes.

2.1. Text Simplification

Text Simplification is an umbrella term that covers any transformation that makes textual information easier to process or understand without changing the meaning of the text. Text simplification is a subset of paraphrasing. Bhagat et al. [5] define paraphrasing as conveying the same meaning using different wording. They go on to describe 25 operations that generate paraphrases. For example, "Synonym substitution" is a paraphrasing operation where we replace words with their synonyms. "Co-referent substitution" is a paraphrasing operation where we substitute a pronoun with the corresponding noun. However, not all these paraphrases lead to simple and more readable texts. In this section, we look at guidelines published by various organizations to simplify text. We also examine the services available to perform this task and the market for simplifying legal texts.

2.1.1. Text Simplification Guidelines

The Plain Language Action and Information Network (PLAIN)[1] is a US-based organization that promotes the use of plain language in government communications. Their guidelines on how to write Plain English text indicate various methods to simplify text. Of the types of paraphrases mentioned in Bhagat et al. [5], "Synonym substitution" and "Verb/Noun conversion" are mentioned as simplifying transformations. Apart from these two, the guidelines also advise writers to avoid or minimize jargon, abbreviations, and definitions.

Plain English Campaign[2] is an effort in the UK to simplify language and prevent miscommunication. They have published a similar set of guidelines on writing in plain language.

[1] `https://www.plainlanguage.gov/guidelines/words/` accessed on 08-10-2021

[2] `http://www.plainenglish.co.uk/how-to-write-in-plain-english.html` accessed on 08-10-2021

These guidelines encourage "Change of voice" and "Verb/Noun conversion" as simplifying transformations (of the ones mentioned in [5]). Apart from them, the guidelines also encourage short sentences, lists, and instructions.

Simple English Wikipedia[3] is a version of Wikipedia where simple words and grammar are used. Simple English Wikipedia's article on how to write articles for them provides yet another set of guidelines on how to write text in Simple English. Their guidelines include the transformations "Change of voice" and "Change of tense" from the list in [5]. Apart from these, they also recommend that the writers avoid jargon or words not present in compilations of Basic English words such as Odgen's BE 850 list[4].

2.1.2. Text Simplification Services

The term 'Plain English' which has gained popularity in the last few years, refers to clear and concise language that avoids the use of uncommon words, jargon, or euphemism. Just like an English language degree, one can now obtain a Plain English Diploma certificate by taking up a 1-year diploma programme which costs approximately £4000. In this course, the participants are taught how to effectively communicate in Plain English[5]. There are a plethora of services out there that employ associates with such training. These services offer 'Plain English' translation services for a fee. Some organizations that offer this service are the National Adult Literacy Agency[6], Polaron[7] and the Plain English Campaign [8]. The costs for this translation range around £150 for 1500 words. There are similar efforts that exist in other languages as well. For example, for the German language, 'Einfache Sprache' or 'Leichte Sprache' are the terms used to refer to the simpler version of the language and there exist paid services offered by companies like Capito GmbH[9] to convert domain specific language to simple, more understandable text.

Another class of text simplification services operate in a more automated manner. For example, services such as Simplish[10] or Online Utility [11] perform lexical simplification. These tools compile the top frequently occurring words in different vocabulary sets like Ogden's Basic English, Top 5000 English words from Wikipedia, top frequent words from the internet, etc. and replace words that do not occur on this list with synonyms from these lists. The rationale behind this method is that more frequently occurring words are more familiar to readers and hence comparatively more comprehensible. These services are free of charge for

[3] `https://simple.wikipedia.org/wiki/Wikipedia:How_to_write_Simple_English_pages` accessed on 08-10-2021

[4] `http://ogden.basic-english.org/wordalph.html` accessed on 08-10-2021

[5] `http://plainenglish.co.uk/services/training/plain-english-diploma.html?trk=profile_certification_title` accessed on 08-10-2021.

[6] `https://www.nala.ie/plain-english/` accessed on 08-10-2021.

[7] `https://www.polaron.com.au/services/naati-language-services/translations/plain-english-translations/` accessed on 08-10-2021.

[8] `https://www.polaron.com.au/services/naati-language-services/translations/plain-english-translations/` accessed on 08-10-2021.

[9] `https://www.capito.eu/` accessed 08-10-2021.

[10] `https://www.simplish.org/` accessed on 08-10-2021.

[11] `https://www.online-utility.org/english/simple_basic_helper.jsp` accessed on 08-10-2021.

smaller texts and charge a subscription fee between 5$ to 25$ per month for larger documents. On the other hand, the text simplification service offered by Lingenio[12] simplifies text based on its grammatical structure. According to the website, the software contains constituency parse based rules that dictate what kind of sentences can be split based on their parse tree. These rules also dictate how to simplify the sentence structure by rearranging the nodes of the parse tree.

None of these automated services are oriented towards the legal domain. However, the existence of services such as Capito GmbH[13] tell us that there is a growing market for such services.

2.2. Corpus creation

In this section, we explore the simple legal texts and existing text simplification corpora out there that we could potentially leverage to create our corpus. We then investigate the existing text alignment techniques that could help us create an aligned corpus.

2.2.1. Simplified Legal Texts

There are a few sources of simplified legal texts on the internet, although the quantity appears quite limited. These sources are not well documented and it is hard to ascertain their quality. For example, Law Easy[14] is a website that explains US Laws in accessible easy to understand videos. Law Rewired[15] is a website that explains different Indian laws in plain English with easy to understand real-life scenarios. The subreddit 'r/legaladvice'[16] where Reddit users post questions or stories asking for legal advice and members of the subreddit offer advice is another source of simple legal text. Another collection of simple legal text is the Law for Dummies collection[17]. Apart from the book series, their website also has step-by-step legal articles explaining the steps required to perform various legal processes. Another example of accessible legal text is the "Simple and Easy Law Info service" (EasyLaw)[18] website provided by the South Korean Ministry of Government Legislation. This website provides accessible legal text divided into various topics and sub topics.

2.2.2. Monolingual Sentence Alignment Techniques

Sentence alignment is the task of finding a mapping between sentences in two documents about the same topic. The information contained in the mapped sentences must be the same. The Neural CRF Model [6] is the current state-of-the-art in sentence alignment. The

[12] `https://lingenio.de/en/services/configuration-of-tools/text-simplifier/` accessed on 08-10-2021.
[13] `https://www.capito.eu/` accessed 08-10-2021.
[14] `https://laweasy.com/` accessed on 08-10-2021.
[15] `https://www.lawrewired.in/` accessed on 08-10-2021.
[16] `https://www.reddit.com/r/legaladvice/` accessed on 08-10-2021.
[17] `https://www.dummies.com/education/law` accessed on 08-10-2021.
[18] `https://www.easylaw.go.kr/CSM/Main.laf` accessed on 08-10-2021.

algorithm performs paragraph alignment first by comparing semantic similarities and relative positioning of the paragraphs and then proceeds to match sentences using sentence similarity criteria. A similar alignment algorithm with a comparable precision is MASSAlign [7] which uses TF-IDF cosine similarity to find starting points in the document to start the sentence alignment and then to perform the sentence alignment itself. Xu et al. [8] use the number of overlapping words to perform sentence alignment between two given articles. In subsection 2.2.2 we discuss the suitability of these methods to our task.

2.2.3. Existing Simplification Corpora

Most of the existing text simplification corpora have been created by aligning Simple English Wikipedia articles with the English Wikipedia articles. PKWP [9] and WikiSmall [10] are one of the first few aligned simplification datasets containing about 100k 1-to-1 as well as 1-to-many aligned simplification samples. Coster and Kauchack [11] created yet another automatically aligned dataset of 137K samples from the same source. The TURK Corpus [12] was the first manually created simplification dataset with 2350 sentences where 8 simplification references per sentence were collected through Amazon Mechanical Turk with 2k of the samples meant for finetuning and the rest for testing. Zhang and Lapata [3] generated another large Wikipedia based dataset with 296K samples known as WikiLarge. Jiang et al. [6] created yet another dataset called WIKI-AUTO (488k samples) using the Neural CRF model. These are some of the notable publicly available datasets created using Wikipedia as a source. Xu et al. [8] compiled yet another simplification dataset, Newsela courpus, from 1130 Newsela articles[19]. In this paper, the corpus is shown to have a better quality compared to the corpora collected from Wikipedia. Another Newsela based dataset called NEWSELA-AUTO (666k samples) has been created using the Neural CRF model [6]. However, neither Newsela based corpora is publicly available and have to be specially requested from Newsela. While we could potentially use these datasets to pretrain transformer models for legal text simplification similar to the BERT-initialized Transformer model trained in [6], this cannot be used to fine-tune the models to work specifically for legal data. These corpora are also unsuitable for studying features of simple legal text since they contain articles from all domains.

2.3. Measuring Complexity of Text

A straightforward manner of measuring how complex a text is, would be to ask an expert to rate how hard the text is on some scale or ask questions from the text and assess the complexity in relation to how fast and how accurately readers can answer these questions. Another popular measurement is the Cloze Test where a few words in the text are blanked out and readers are asked to guess them [13]. However, these are time consuming and manual tasks which means we require some other machine calculable features which are correlated with ease of reading and understanding text.

[19] `https://newsela.com/` accessed on 08-10-2021

An easy indicator of text complexity would be the number of 'difficult words' used in the text. The python package textstat[20] computes this by looking up the number of words that don't appear in a list of easy words[21]. Sentence length and other length-based metrics are also good indicators of text simplicity [14]. Some of the commonly used length-based readability formulae which use easily computable features such as sentence length, word length, number of syllables etc. are the Flesch Kincaid grade [15], Gunning fog index [16], SMOG index [17], Automated Readability Index [18], The Coleman-Liau Index [19], Linsear Write Formula [20] and Dale-Chall Readability Score [21]. These are only a handful of more than 200 such formulae as mentioned in [22], but we choose these readability formulae to analyze our generated corpus due to their ease of use. These metrics can all be computed using the python library 'textstat'. We discuss the formulae in detail and interpret the scores in section 4.2.

Some interesting readability formulae that take grammatical structures such as number of prepositions and other complex syntactical properties are the Winnetka Readability formula [23] and the Coh-Metrix [24]. Another visual metric is the Fry readability formula [25] which plots syllables vs number of sentences in a text onto a graph marked with areas of complexity. Some other proxies for measuring how complex a piece of text is, would be to measure the reading speed by tracking the reader's eye movement [26], by monitoring changes in the brain when the text is being read [27] or by identifying words or phrases that take a longer time to read aloud [28] but these methods involve time-consuming setup and measuring processes.

2.4. Automated Text Simplification Techniques

Al-Thanyyan et al.'s survey of automated text simplification [29] contains an extensive list of the automated text simplification methods that exist in literature up to early 2020. They broadly divide these methods into 4 categories - lexical approach, syntactic approach, machine translation and hybrid approaches. The methods in the lexical approach attempt to replace the complex words with simple ones, i.e., they perform lexical simplification. The other approaches attempt to change the structure of the text as well as the words in order to make it easier to read, i.e., they attempt to perform paraphrasing to simplify text; this is also just known as (normal) text simplification. The LSBert mode [30] appears to be state-of-the-art in performing lexical simplification with its F1-Score outperforming the existing lexical simplification models in all lexical simplification benchmarks. In the normal text simplification task, the MUSS: Multilingual Unsupervised Sentence Simplification by Mining Paraphrases [2] model outperforms the other models on the two most recent publicly available benchmarks ASSET [31] and TurkCorpus [12]. We examine how well these models simplify legal text in chapter 5.

[20] `https://pypi.org/project/textstat/` accessed on 08-10-2021

[21] `https://github.com/shivam5992/textstat/blob/1693bf0d9dcc451896955e1fe4604402cfe5a968/textstat/resources/en/easy_words.txt` accessed on 08-10-2021

3. Corpus Creation

As discussed in chapter 2, there exist some text simplification datasets. Most of these datasets are created by aligning Wikipedia articles. However, these articles do not belong to the legal domain which makes them rather unsuitable for studying the characteristics of simple legal text. Moreover, a study by Xu et al.[8] shows that these datasets tend to be noisy and that many of the samples in the dataset are not simplifications - they are deletions or other modifications that do not count as simplifications. They also do a manual study and claim that only 12% of the samples are pure simplifications. Creating a corpus of easy legal text and aligning it with the original text has multiple benefits. The main benefit, in the purview of this thesis, is that we can study the characteristics of simple legal text and provide guidelines on how to generate simple legal text. This could then be used to generate a training dataset for legal text simplification or as an evaluation dataset for such a task. This section discusses our methodology of creating an aligned dataset of simple and standard laws.

3.1. Sources of easy and standard legal text

In this thesis, we consider various online sources of simple legal text for creating the legal text simplification dataset.

The first source we considered is the Law for Dummies[1] collection as this is an established book series which contributes to the credibility of this data source. The collection of articles online comprised of around 220 legal articles with various topics explained in a simple manner. However, in order to present the content in a simple manner, many of the articles are written as instructional guides and it is hard to distinguish generic advice from the legal statements and align them with standard laws.

Another good source of simple legal text would be the subreddit 'r/legaladvice'[2] where reddit users post questions or stories asking for legal advice falling under various categories such as Labour Law, Housing Law, Criminal Law etc. and members of the community offer advice on how to approach the issue. The subreddit appears to offer legitimate and useful legal information, especially regarding which authorities to approach for different kinds of situations, however there is an extensive lack of references to the original law or any official source of information. This makes it hard to collect sources of the original law.

Law Rewired[3] is another website that presents Indian Laws in a simple to understand manner. This is a potentially good source of simple legal text as it explains various acts and

[1] `https://www.dummies.com/education/law` accessed on 08-10-2021.
[2] `https://www.reddit.com/r/legaladvice/` accessed on 08-10-2021.
[3] `https://www.lawrewired.in/` accessed on 08-10-2021.

articles falling under 9 different sections of laws. However, some of the characteristics of this source make it difficult to extract and align the text with the corresponding law it explains. Hence we would require extra intelligence, in the form of a classifier, to detect whether a sentence is a part of this story or an explanation of a statute. Moreover, the articles also do not provide any reference to the original laws which makes it difficult to find the corresponding laws or validate any alignment.

Finally, all the sources mentioned so far are crowdsourced or simplified by third-parties, which means that it is difficult to ascertain the accuracy of the information. The South Korean Ministry of Government Legislation has published a website[4] (hereon referred to as EasyLaw) which they describe as a "Simple and Easy Law Info" service.

This effort is unique with respect to its origin, that is, it is an effort from the official source of laws to present them in a simplified manner. The foremost advantage of this source is that its authenticity is assured. The second advantage of this source is that extensive references to the original laws are provided. Moreover, the simple explanations of different laws are provided under sections and subsections grouped under practical headings, but without any additional embellishments like stories or personal comments. The standard South Korean laws are also available online at the website of the Korean Law Translation Center[5] (hereon referred to as E-Law). These features of the simplified legal text on the EasyLaw website make this data source a good candidate to create an aligned dataset of simple vs. standard law from.

3.2. Scraping and storing the data

In this section, we discuss the structure of the websites and the strategy used to scrape the text from the EasyLaw as well as the E-Law website. We also describe the manner of and the rationale behind the chosen storage format.

3.2.1. Organization and User Interface of the EasyLaw website

The EasyLaw website has simple South Korean Laws translated into 12 different languages. We concentrate on the English language text first as the English language audience is currently the largest on the internet[6]. The content of the EasyLaw website is divided into broad topics such as Real Estate, Family Law, All, etc. (Level 1 on Figure 3.1). Under each of these topics, the authors have identified the main problem areas or commonly searched for information and organized them into multiple sub-topics (Level 2 on Figure 3.1).

Selecting one of these sub-topics takes one to an page describing the general content as well as the key sections covered within this sub-topic. Finally, choosing one of the sections (Level 3 on Figure 3.1) brings us to the page with the simplified legal text. Here too, the text is presented over various sections and subsections, navigable via a sidebar as seen in

[4] `https://www.easylaw.go.kr/CSM/Main.laf` accessed on 08-10-2021.

[5] `https://elaw.klri.re.kr/eng_service/lawView.do?hseq=37688&lang=ENG` accessed on 08-10-2021.

[6] `https://www.internetworldstats.com/stats7.htm` accessed on 08-10-2021

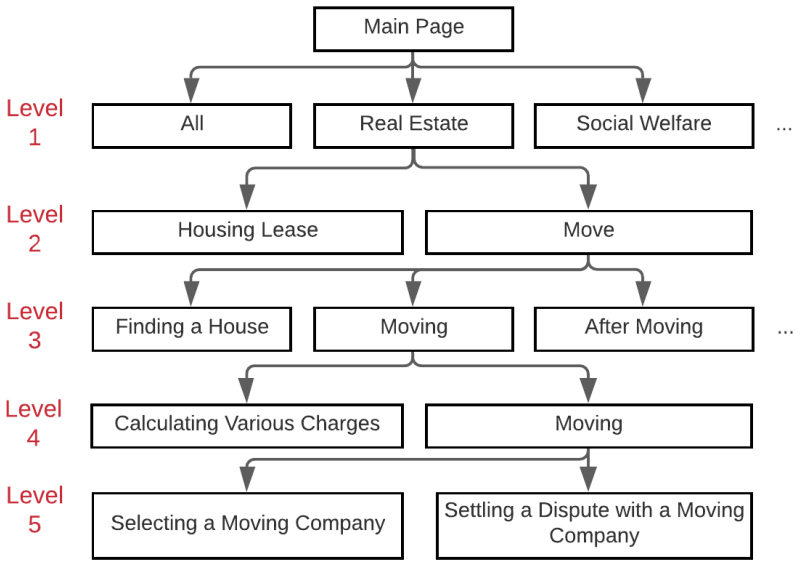

Figure 3.1.: Organization of the Easy Law website. [7]

Figure 3.2. The sidebar allows us to choose subsections (Level 4) under each Level 3 section and the articles containing the EasyLaw text (Level 5) are listed under the Level 4 subsections.

The main text we want to scrape and align is present at Level 5. We must navigate the domain from the main topics page in order to collect all the articles to maximise the amount of data collected.

3.2.2. Scraping the EasyLaw website

Throughout this section, we using the python library `urllib.request` to fetch pages. We then use the python package BeautifulSoup [32] to navigate and extract data from the page.

We navigate to the English EasyLaw website and select the category 'All' (Level 1) since we want to scrape all the articles. Our next step is to extract a list of all subtopics (Level 2 links) from the page. Upon examining the page and its Document Object Model (DOM) tree (example in appendix Figure B.2), we observe that in order to extract the link to all the articles, we must extract the `href` attribute values which contain the pattern `SubCcfCmd.laf`. Moreover, we see that the article links are displayed in a paginated manner where only 3 article links are displayed on the page. We next examine the Network panel on Chrome Developer tools and notice that the page employs server side pagination when the next page is requested. The Network panel also shows us the parameters that need to be passed in order to retrieve the next page. The parameter `targetRow` is incremented by 3 for every next page and there are totally 47 article subtopics. We use this parameter to retrieve each page and then use the URL pattern `SubCcfCmd.laf` to extract and store all subtopic links. We also

[7] `https://www.easylaw.go.kr/CSM/SubCnpclsCmd.laf?csmSeq=892&ccfNo=3&cciNo=1&cnpClsNo=2` accessed on 08-10-2021

introduce a sleep timer of 5 seconds between consecutive requests to prevent our script from overloading the EasyLaw server.

Now, we have scraped links to all subtopic pages (Level 2) and we see that each subtopic page is identified by a unique `csmSeq`. We observe that clicking each of the links on the subtopic page leads us to different articles but all these articles have the same sidebar (Figure 3.2). Hence, for each subtopic, we must navigate to any of these pages with the subtopic sidebar and then scrape all the links in the sidebar to obtain links to all the EasyLaw articles. If we examine the Network panel, we see that on selecting a link on the subtopic page, 3 additional parameters are being set to fetch the article, namely, `ccfNo` which refers to the sections in an article (Level 3), `cciNo` which refers to the subsections in the section (Level 4), and `cnpClsNo` which refers to the modules under each subsection (Level 5). Hence, each page with text is uniquely identified with a combination of `csmSeq`, `ccfNo`, `cciNo` and `cnpClsNo`. We have scraped all existing subtopics or all possible `csmSeq` ids; however, we still need to find the existing combinations of the other ids. Since every subtopic has at least 1 section, subsection and module, we can retrieve the first page of the subtopic by assigning `csmSeq=1`, `ccfNo=1`, `cciNo=1` and `cnpClsNo=1`.

We then extract the links to all article pages under the subtopic from the DOM of the sidebar on this page. A quick inspection of the DOM tree tells us that the relevant links contain a pattern that matches the string `SubCnpclsCmd`. We scrape all links that match this pattern. We repeat this process for all known values of `csmSeq`. Using this list of links, we may now retrieve all the relevant pages.

Figure 3.2.: EasyLaw[8]article page sidebar to query parameter mapping.

We now have URL references to all the articles that contain the simple legal text we are interested in. Subsection 3.2.5 discusses how this data is processed, modified and stored.

[8] `https://www.easylaw.go.kr/CSM/CsmOvPopup.laf?csmSeq=1581&ccfNo=1&cciNo=1&cnpClsNo=1` accessed on 08-10-2021

3.2.3. Organization and User Interface of the E-Law website

The website containing the translated standard laws is organized in a more straightforward fashion. It provides a search page to search for all Acts. The sidebar in Figure 3.3 allows for navigation between different categories of Acts and clicking on the category provides the list of all Acts under the category. Selecting the Act redirects the user to a page containing the text of the entire Act. We need to scrape all such pages from this domain.

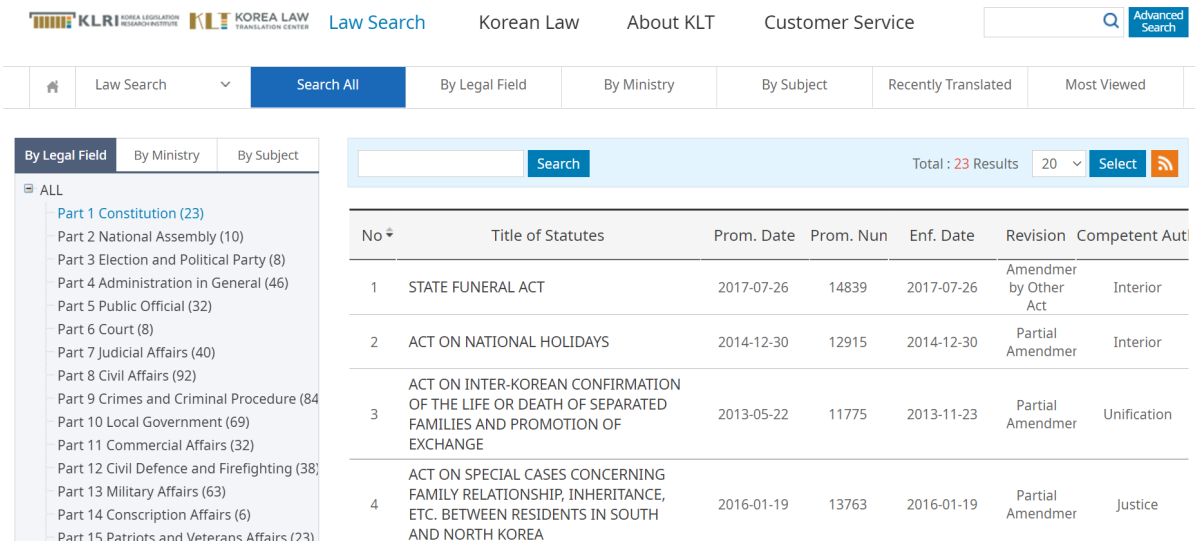

Figure 3.3.: Screenshot of the E-Law Law Search Service [9]

3.2.4. Scraping the E-Law website

Examining the Chrome Developer Console for requests sent on selecting a category in the navigation sidebar in Figure 3.3 shows us that a request is sent to retrieve all Acts that fall under this category. In the request, we see two parameters that we can modify to obtain all Acts in the same request, namely, `listCnt` and `rows` which indicate the number of Acts to be fetched and shown. A quick glance through the categories shows that the maximum number of Acts under any particular category is 134. Hence we can set these parameters to 134. Another parameter which is useful for our use-case is `pCode` which refers to the category from 1 through 44. We can vary this parameter through all possible values to get the metadata of all the Acts. The response to this request is a JSON which contains some metadata about each Act in this category. In this metadata JSON, the key `historySeq` is a unique identifier for each Act and hence we extract and store all `historySeq` values. This allows us to construct the URL for all E-Law acts by substituting `historySeq` in `https://elaw.klri.re.kr/eng_service/lawView.do?hseq=historySeq&lang=ENG`.

[9] `https://elaw.klri.re.kr/eng_service/lawAllSearch.do?pCode=01` accessed on 08-10-2021

3.2.5. Storing the data

Once the addresses to the required texts have been scraped, there are multiples options to process and store the text from these pages. To understand why the manner of storing the data is important, let us look at an example use case. Figure 3.4 is an excerpt from the EasyLaw texts, where the information given in the numbered list falls under the first line of text. This means, the entire excerpt refers to one law and are not standalone bits of information. As a reader, this is easy to recognize due to the formatting.

- The term "**investment object**" means investments made by a foreign investor to possess stocks or st akes and refers to any of the following (Subparagraph 8 of Article 2 (1) of the Foreign Investment Pro motion Act and Article (10), (11), and (12) of the Enforcement Decree of the Foreign Investment Prom otion Act).

① Means of foreign payment under the Foreign Exchange Transactions Act or means of domestic payme nt by the exchange thereof

② Capital goods

③ Proceeds from the stocks or stakes acquired under the conditions as prescribed by the Foreign Invest ment Promotion Act

④ Industrial property rights, the right to be utilized in the industrial activities from among the copyright s under the Copyright Act, and the layout-design right under the Act on the Layout-Designs of Semic onductor Integrated Circuits, other equivalent technologies thereto, and rights pertaining to the use o f such rights or technologies

Figure 3.4.: Sample EasyLaw excerpt describing the definition of "investment objects"[10]

However, the structure of DOM tree does not suggest this grouping. As we can see in Figure 3.5, the two *div*s are not structured or nested, but are siblings in the DOM tree. This does not directly provide any information regarding the sections and subsections in the text and hence, we cannot recognize which texts belong together and which texts are standalone.

The font based or visual grouping of texts in the HTML pages, must, therefore be translated into a nested data structure that retains the semantic structure of the legal text. This can be done by translating the serial HTML `divs` into nested `divs` or into nested python dictionaries. We choose to translate the HTML to python dictionaries due to its visual as well as programmatic simplicity.

The code used to perform this translation for EasyLaw and E-Law can be seen in section A.1 and section A.2 respectively. The resulting dictionary for the excerpt in Figure 3.4 can be seen in Figure 3.6. We fetch and store 717 EasyLaw articles and 2183 E-Law Acts this way.

3.3. Aligning easy and standard data

Now that we possess a list of simple legal articles and a list of Acts, we align them such that each standalone piece of simple legal information is mapped to one or more corresponding

[10] `https://www.easylaw.go.kr/CSM/CsmOvPopup.laf?csmSeq=1581&ccfNo=1&cciNo=1&cnpClsNo=1` accessed on 08-10-2021

```
▼<div id="div1581.1.1.1.4325500" class="plv3b" style="text-align:justify;"> == $0
    "- The term “”
    <strong>investment object</strong>
    ”” means investments made by a foreign investor to possess stocks or stakes and refers to any of the following (Subparagraph 8 of Article 2 (1) of the Foreign Investment Promotion Act and Article (10), (11), and (12) of the Enforcement Decree of the Foreign Investment Promotion Act)."
  </div>
  <div class="spaceDiv2"></div>
▼<div id="div1581.1.1.1.4325501" class="plv3d" style="text-align:justify;">
    "① Means of foreign payment under the Foreign Exchange Transactions Act or means of domestic payment by the exchange thereof"
  </div>
```

Figure 3.5.: DOM tree of EasyLaw excerpt as shown in Figure 3.4

legal Acts or Articles providing the same information. To formulate this more precisely, given a list of simple articles S_1, S_2, S_3,... with article S_n containing sentences $s_{n,1}$, $s_{n,2}$, $s_{n,3}$,... and a list of Acts A_1, A_2, A_3,... with Act A_m containing sentences $a_{m,1}$, $a_{m,3}$, $a_{m,3}$,... we want to find a mapping of

$$
\begin{aligned}
&s_{n,1}, s_{n,2}, s_{n,3}, \ldots, s_{n,m} \\
&\Updownarrow \\
&a_{x,i}, a_{x,i+1}, \ldots \\
&\vdots \\
&a_{z,k}, a_{z,k+1}, \ldots
\end{aligned}
$$

such that the information contained on both sides of the mapping is the same. Here $s_{n,1}$, $s_{n,2}$,..., $s_{n,m}$ corresponds to a standalone piece of simple legal text and each $a_{x,y}$ corresponds to a sentence from an original Act. This involves recognizing the standalone piece of information, finding the Acts that this information refers to and finally extracting the corresponding sentences from the identified Acts.

3.3.1. Limitations of existing alignment techniques

As discussed in subsection 2.2.2, there are quite a few techniques that perform monolingual sentence alignment. However, these techniques require as input, two articles that contain the

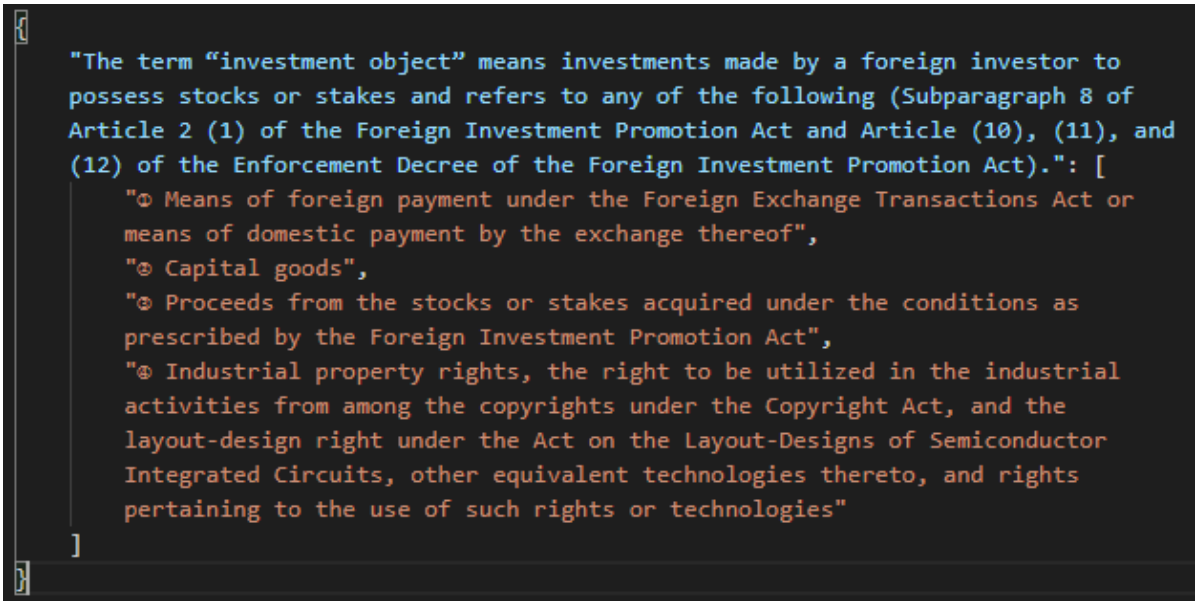

Figure 3.6.: Excerpt translated from Figure 3.4 into JSON format

same information about the same topic. For example, the Neural CRF model [6] can align sentences within two Wikipedia articles with the same title.

That is to say, they perform mappings of the form

$$s_{n,1} \Updownarrow a_{x,1}$$

Here the information unit is fixed as 1 sentence and the articles n and x are already known and the mapping has to be performed within these articles. The task we are attempting to perform is a super-set of the conventional sentence alignment tasks. Where conventional models attempt to align one sentence from one document with another sentence from a single known document, we need to align it with text from multiple documents since a single EasyLaw text could refer to information from multiple standard Acts (An example of this can be seen in Figure B.1). Hence we cannot directly perform Monolingual Sentence Alignment using any of the existing models and must introduce our own techniques to do perform this task.

3.3.2. Selecting and extracting information from source text

The first task is to identify standalone pieces of legal information in the EasyLaw texts or the source text. In the EasyLaw texts, each standalone piece of information seems to have an associated reference to the original Act(s) as seen in Figure 3.7. This also tells us where to

look for the corresponding information. This reference mentions the name of the Act and the Article which have been presented in a simplified manner in the text.

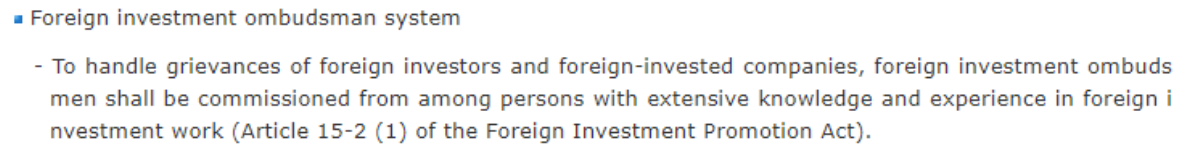

Figure 3.7.: A standalone piece of legal information from the EasyLaw website[11]

We look for the pattern in which these references are presented. Some sample patterns are seen in Figure 3.8.

(Proviso to Article 26 of the Labor Standards Act, Article 4 and attached table of Enforcement Rules of the Labor Standards Act)

(Article 15.(1) of the Foreign Investment Promotion Act)

(Article 15-2.(1)&(2) of the Foreign Investment Promotion Act)

(Subparagraph 5 of Article 22(3) of the Standardized Agreement of the Door-to-Door Delivery)

(Article 3(2) of the Product Liability Act)

(Article 51(2)20 of the Electrical Appliances and Consumer Products Safety Control Act)

(Article 2(11)(a) of the Electrical Appliances and Consumer Products Safety Control Act and Article 3(3) of the Enforcement Rule of the Electrical Appliances and Consumer Products Safety Control Act)

(Subparagraph 2 of Article 34(3) of the Act on the Regulation of Terms and Conditions)

Figure 3.8.: Patterns of E-Law Article references in EasyLaw articles[12]

We formulate the most common patterns into a regular expression that represents it. A regular expression is a series of symbols that represents a pattern. This expression is then used to search strings for the pattern it represents. We craft a regular expression to represent all the patterns of Article information that start with '(Article' and search over all the collected EasyLaw texts. We then extract any matched text along with all its children, that is, the sections and subsections that come under it as seen in Figure 3.9. The code to perform this

[11] `https://www.easylaw.go.kr/CSM/CsmOvPopup.laf?csmSeq=1581&ccfNo=4&cciNo=4&cnpClsNo=1` accessed on 08-10-2021

[12] `https://www.easylaw.go.kr` accessed on 08-10-2021

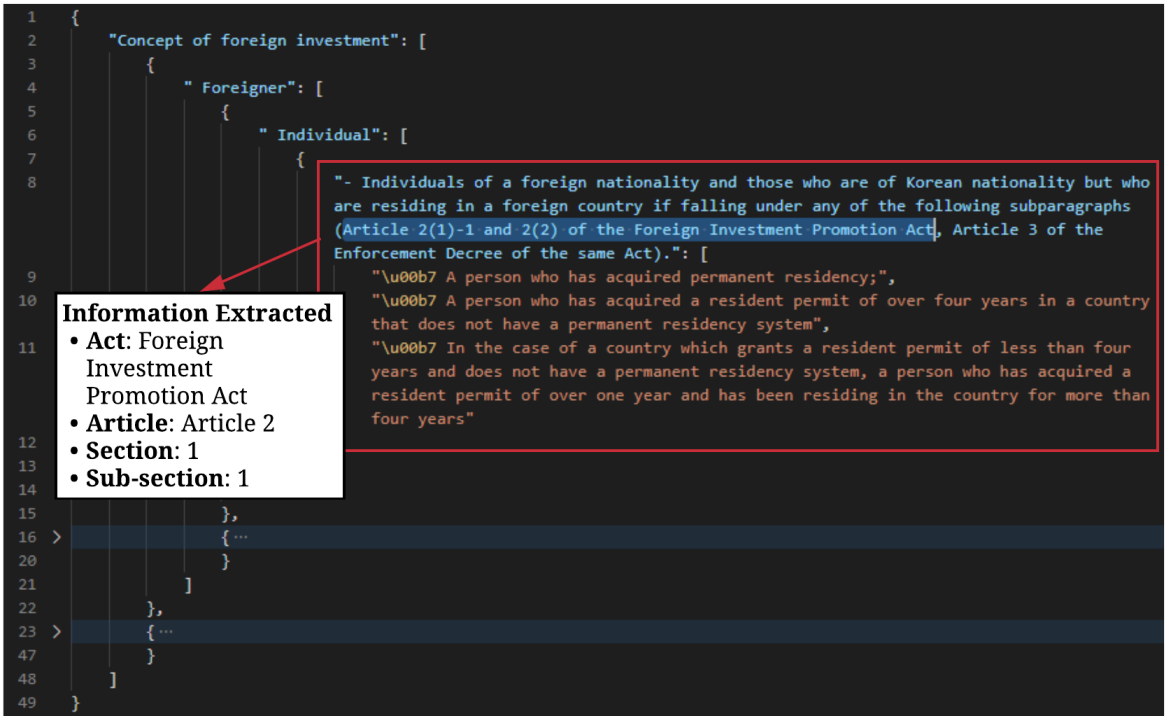

Figure 3.9.: Information extracted via Regex match from EasyLaw json document.

match can be seen in section A.3. This selected object is then condensed into text form and this forms one unit of simple legal information.

3.3.3. Match source with target

Once we have the source text and the information regarding the associated Act(s), we now try to find the target text that matches the information in the source text.

First we organize the information extracted from the E-Law website[13] (translated South Korean Acts) so that it's faster to search through. To do this we index all the Acts by the Act Name. We create a dictionary to lookup the path of the Act file using the Act Name. Searching through the data directly involves file operations which take time; moreover this would be a linear search. However, a reverse dictionary lookup can be done in constant time. This drastically reduces the time required for lookup by a factor of at least 2000.

Next we try to find a match for the Extracted Act Name in our index. If a direct match is found, we then attempt to find the next level of information within this file. The next level in this case would be Article number which can always be found as a direct match. Similarly we find the smaller levels of sections and subsections as seen in the example Figure 3.10. Figure 3.10. An interesting challenge here is that since both the simple laws as well as the standard Acts have been translated independently, there often isn't a direct match. Some

[13] `https://elaw.klri.re.kr/eng_service/lawView.do?hseq=37688&lang=ENG` accessed on 08-10-2021.

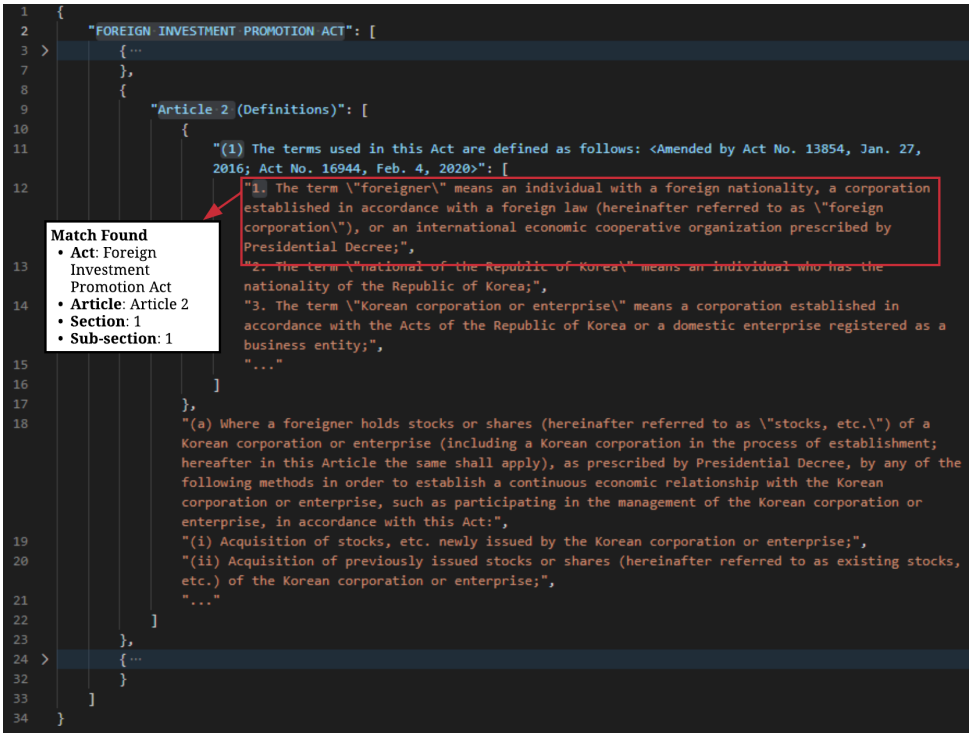

Figure 3.10.: Information extracted in Figure 3.9 matched with E-Law document.

examples of Act names that refer to the same Act but have different text can be seen in Table 3.1

Table 3.1.: Examples where extracted Act Name differs from actual Act Name

Extracted Act (EasyLaw)	Actual Act (ELaw)
CIVIL LAW ACT	CIVIL PROCEDURE ACT
CIVIL PROCEEDINGS ACT	CIVIL PROCEDURE ACT
STATE PROPERTIES ACT	STATE PROPERTY ACT
LOST ITEMS ACT	LOST ARTICLES ACT
ENFORCEMENT RULE OF THE PASSPORT ACT	ENFORCEMENT DECREE OF THE PASSPORT ACT

To solve this problem, we use the BERT language model [33] to compare the two Act Names. BERT is a language model that is capable of producing a sentence embedding given a piece of text. This sentence embedding is a vector of numbers and provides a numerical representation of the meaning of the text. The idea here is that two pieces of text which have a similar meaning will have sentence embeddings that are at a smaller cosine distance from each other that texts which have different meanings. In Table 3.2 we observe that the first two rows where the Act Names are the same, but have spelling errors, the model gives

them embeddings that are close to each other, hence the lower cosine distance. However, in the third example, the Act names refer to two different acts and hence hold two different meanings and have a higher cosine distance.

Table 3.2.: Cosine distance between similar and dissimilar Act Names

Extracted Act (EasyLaw)	Actual Act (E-Law)	BERT cosine distance
ENFORCEMENT DEGREE OF THE FRAMEWORK ACT ON CONSUMERS	ENFORCEMENT DECREE OF THE FRAMEWORK ACT ON CONSUMERS	0.006135029444
ENFORCEMENT DCREE OF THE FRAMEWORK ACT ON CONSUMERS	ENFORCEMENT DECREE OF THE FRAMEWORK ACT ON CONSUMERS	0.006404538728
ENFORCEMENT RULE OF THE ACT ON THE ENFORCEMENT	ENFORCEMENT DECREE OF THE FRAMEWORK ACT ON CONSUMERS	0.008788731672

While, in theory, this helps us find a match by looking for a match with the lowest cosine distance, in practice, it is also possible that there is no match. Hence, we need to set a threshold score, such that $cosinedistance < threshold_{bert}$ would imply a match - otherwise we fail to find a match for the Act.

However, a $threshold_{bert} > 0$ which also includes matches other than an exact match includes a risk of including erroneous matches, i.e., a risk of false positives. Increasing the $threshold_{bert}$ increases the number of matches as seen in Figure 3.11.

However, if we do an evaluation of the accuracy of these matches, we see that the number of errors also increases drastically as the $threshold_{bert}$ increases (Figure 3.12).

If the data is to be used for tasks that can tolerate some amount of erroneous data, we could potentially use thresholds higher than 0.002. However, for the purposes of the rest of this thesis, we use a threshold of 0, so that we can perform the studies and answer the other research questions with a higher confidence.

3.4. Discussion

Using the techniques discussed in this chapter, we are able to generated a structured corpus of simple legal text, standard legal text as well as aligned pairs of (simple, standard) legal texts. As we saw in subsection 3.3.3, attempting to match the simple and standard Acts using BERT embeddings did not improve our recall without hurting precision. A potential way to improve this would be to use a BERT model trained on legal data. It is possible that since the generic BERT model is trained on text from Wikipedia and Book Corpus [33], it assigns similar embeddings to all legal texts and hence is unable to distinguish texts within the legal domain. However, if we use a version of this model trained on legal text already [34], the

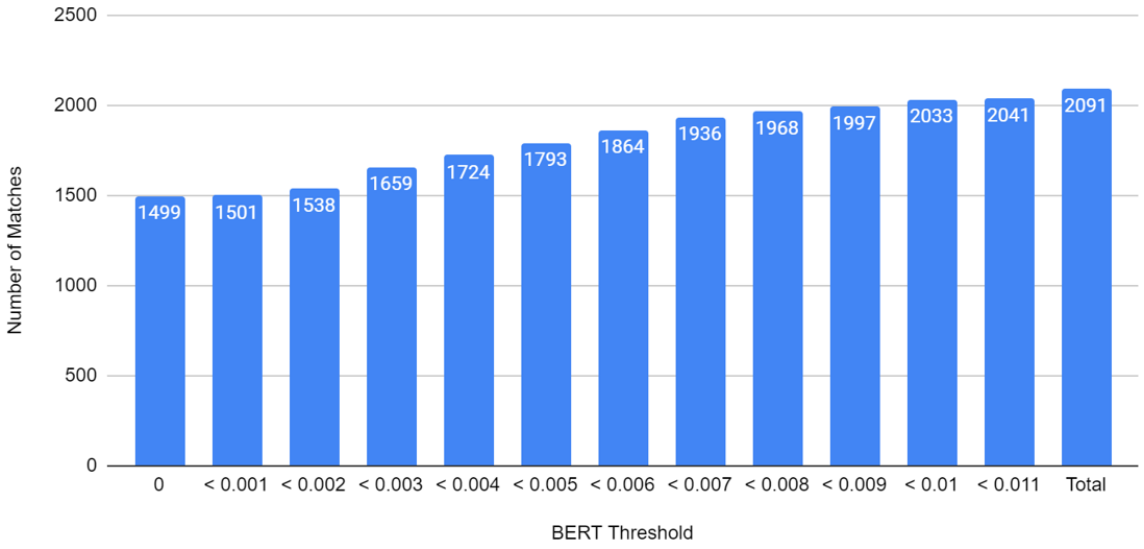

Figure 3.11.: Number of EasyLaw and E-Law Matches vs the BERT Threshold. Increasing the $threshold_{bert}$ increases the number of matches.

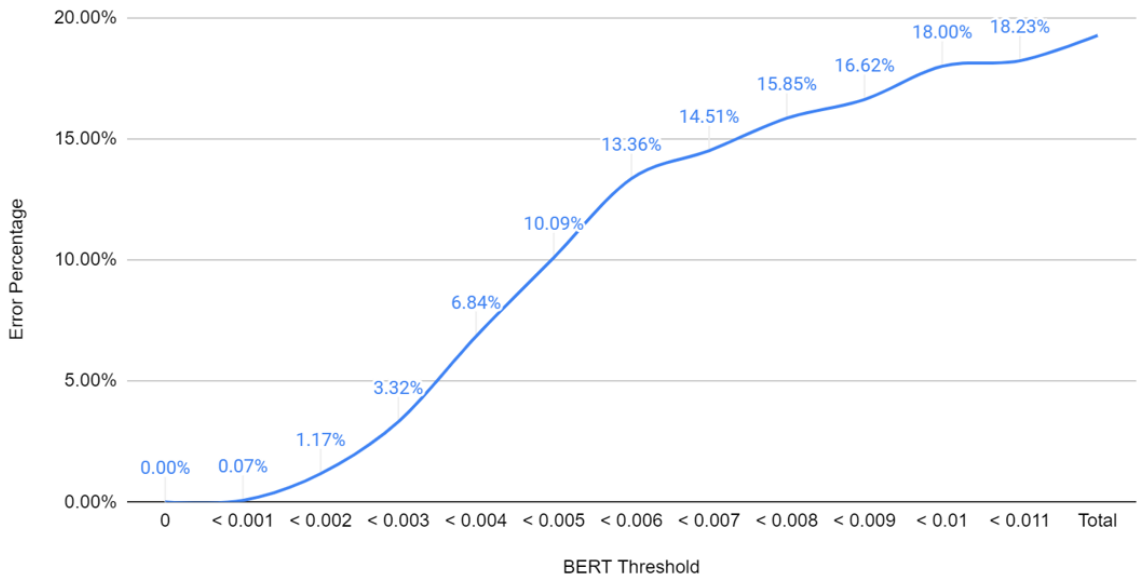

Figure 3.12.: Number of Erroneous Matches vs. BERT Threshold. Number of errors also increases drastically as the $threshold_{bert}$ increases.

model maybe able to distinguish texts within the legal domain better and we may be able to improve the recall without hurting precision.

Currently, in the aligned corpus, we see many samples where the standard law we select using the article information extracted from the EasyLaw text contains more information than the EasyLaw text itself. We could refine the dataset further by only picking out those sentences from the selected E-Law article where the information corresponds to the information in the EasyLaw text. For this purpose we may now use the alignment techniques mentioned in subsection 2.2.2 as the limitations discussed in subsection 3.3.1 no longer apply once we have done the alignment of EasyLaw text to E-Law article.

4. Characteristics of Accessible Legal Text

On an absolute scale, without performing any comparisons, there are many ways to characterize and measure different aspects of the simplicity of textual content. One can measure the readability of text using various readability scores that measure the grade level required to read a piece of text. Another aspect of text that can be measured is the structure. The structure of text can indicate the grammatical structure or the schematic structure. Measuring frequencies of various grammatical structures can indicate the grammatical complexity and variety of writing styles used. Another interesting feature to study is the schematic structure and formatting of text to understand if there is a significant difference in the way they influence comprehension. A feature that is unique to legal text is the extensive referencing that links one part of the text to another. The larger the degree of reference, the more the reader must remember, for context, and the harder text may become to comprehend. Analyzing this feature conveys the degree of the effect that extensive referencing could have on the readability of legal text.

4.1. Count based complexity measures

Count based readability statistics compares how understandable text is, on the basis of counts of different linguistic features. An intuitive example of a count based statistic would be the average syllable count per line of text. A larger syllable count indicates either long sentences or large words both of which could potentially make text harder to comprehend.

4.1.1. Computing the scores

We compute the count based readability scores using the python textstat[1] package. This package provides functions such as `syllable_count`, `lexicon_count`, `sentence_count` and `difficult_words`. `syllable_count` uses Hunspell hyphenation dictionaries[2] which returns all possible hyphenation positions each of which usually occurs after a syllable. `lexicon_count`, `sentence_count` are the number of words and sentences respectively, in the text. `difficult_words` returns the number of words not present in textstat's list of easy words[3]. We compute these metrics on each data sample from the aligned courpus to create a box-plot to visualize these metrics Table 4.1.

[1] `https://pypi.org/project/textstat/` accessed on 08-10-2021

[2] `https://github.com/the-happy-hippo/pyphen` accessed on 08-10-2021

[3] `https://github.com/shivam5992/textstat/blob/1693bf0d9dcc451896955e1fe4604402cfe5a968/textstat/resources/en/easy_words.txt` accessed on 08-10-2021

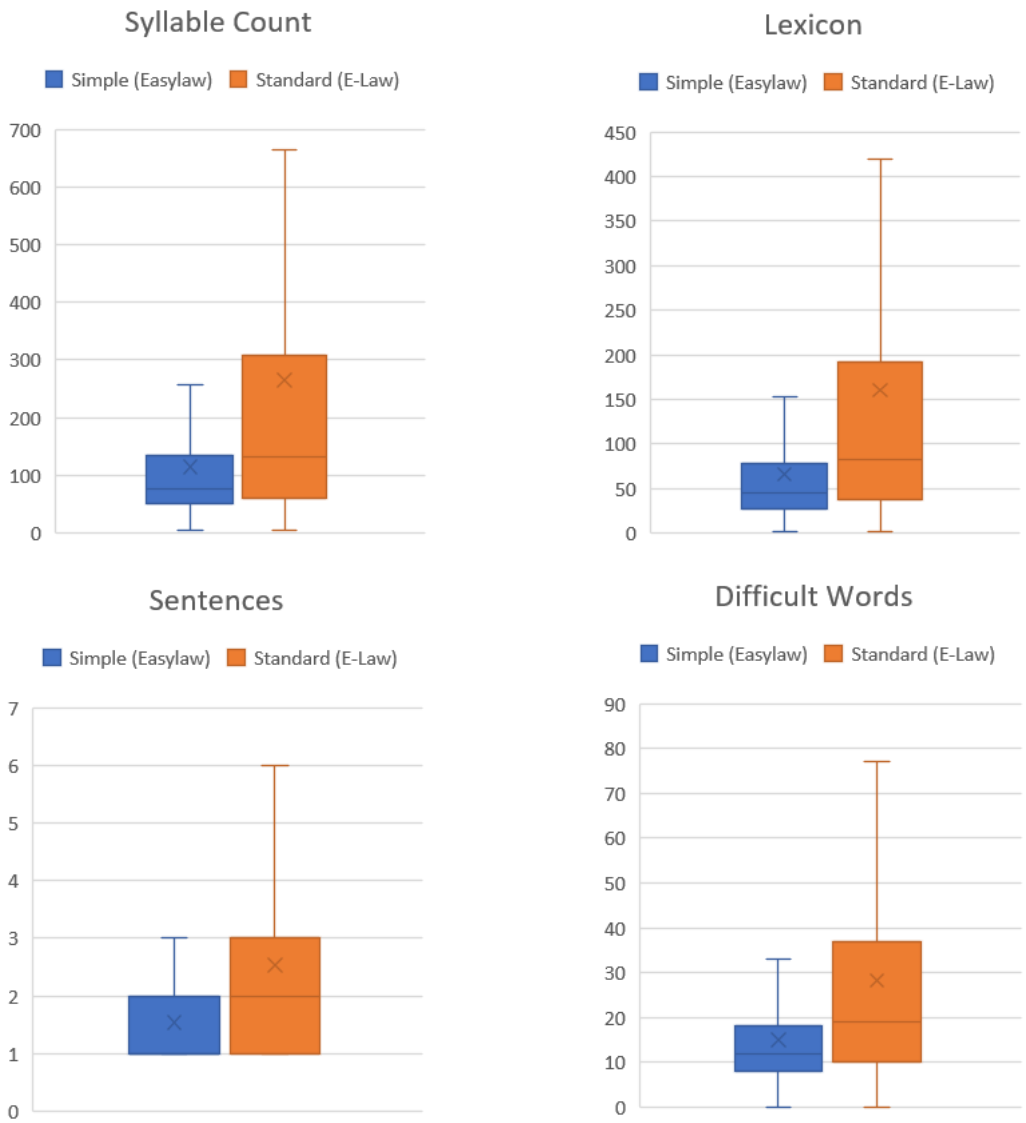

Table 4.1.: Count based readability scores of EasyLaw vs ELaw texts.

4.1.2. Interpretation

As we see in Table 4.1, the mean syllable count for the E-Law text seems much higher, indicating that E-Law sentences have either larger words or larger sentences.

The Lexicon count refers to the number of words per text sample. A higher count could correlate with higher chances of text containing unfamiliar words, which hampers the readability of the text. In this case too, Table 4.1 shows us that E-Law texts tend to have a higher average lexicon count.

A third measure of difficulty is the number of sentences. It's hard to establish a direct correlation between the number of sentences and readability since a larger number of sentences could indicate less readability due to unnecessary information or redundancy but it could also indicate a more readable text since the same information could be explained in more but shorter sentences. The E-Law and EasyLaw sentences seem to have a similar median number of sentences but E-law texts seem to have a higher average number of sentences. This could either imply that the E-Law text has more information than EasyLaw or that E-Law has shorter but more sentences with the same information.

The fourth and final count measure is easier to interpret. The difficult word count looks at the number of words in the text that don't appear on an easy word list. This list is usually compiled by collecting the frequent-most words from a text source such as Wikipedia, the internet, etc. The correlation here is direct - if the number of uncommon words in a sentence is higher, it is more likely that the reader may come across many unfamiliar words and be unable to understand the text. Hence the complexity of the text is likely to be higher. Table 4.1 clearly shows that E-Law texts have a higher average as well as median count of difficult words.

4.2. Length-based complexity measures

Grade-based readability scores are all length-based complexity measures that indicate the grade of schooling or education one needs to have in order to understand a particular piece of text. Flesch Kincaid grade [15], Gunning fog index [16], SMOG index [17], Automated Readability Index [18], The Coleman-Liau Index [19], Linsear Write Formula [20] and Dale-Chall Readability Score [21] are all grade-based readability scores. Different grade-based scores depend on different linguistic features of text such as presence of difficult words or average word length among others. Each grade based score gives a different weight to these features and by looking at the comparative scores, we can gain some insight on how the features of the two texts are skewed.

4.2.1. Computing the scores

We compute the count based readability scores using the python textstat[4] package. This package provides functions such as

[4] `https://pypi.org/project/textstat/` accessed on 08-10-2021

- `flesch_kincaid_grade`,
- `gunning_fog`,
- `automated_readability_index`,
- `smog_index`,
- `coleman_liau_index`,
- `linsear_write_formula`,
- `dale_chall_readability_score`,
- `text_standard` (this computes the readability consensus), and
- `flesch_reading_ease`.

As seen in Equation 4.1 the `flesch_kincaid_grade` function uses a weighted sum of number of words per sentence and number of syllables per words. Since number of words per syllable has a higher weight, large words or words with more syllables contribute to a higher complexity according to this metric.

$$\text{Flesch Kincaid Grade} = 0.39\left(\frac{\text{total words}}{\text{total sentences}}\right) + 11.8\left(\frac{\text{total syllables}}{\text{total words}}\right) - 15.59 \tag{4.1}$$

As seen in Equation 4.2 the `gunning_fog` function uses the weighted sum of number of words per sentence and the percentage of complex words. Since the percentage of complex words and average number of words per sentence are equally weighted, this metric indicates an equal effect of both these components on text complexity.

$$\text{Gunning Fog Index} = 0.4\left[\left(\frac{\text{words}}{\text{sentences}}\right) + 100\left(\frac{\text{complex words}}{\text{words}}\right)\right] \tag{4.2}$$

As seen in Equation 4.3 the `smog_index` function calculates a metric solely based on the number of polysyllable words (here, polysyllable refers to 3 or more syllables) per sentence. In this metric, this parameter is the sole indicator of complexity. However, the `smog_index` is only computed if the text has at least 3 sentences, else the `smog_index` value is 0.

$$\text{SMOG Index} = 1.0430\sqrt{\text{number of polysyllables} \times \frac{30}{\text{number of sentences}}} + 3.1291 \tag{4.3}$$

The `automated_readability_index` function (Equation 4.4) is a weighted sum of the average word length in characters and average number of words per sentence. According to this metric average word length in characters contributes more to the complexity of the text as compared to the length of the sentence itself.

$$\text{Automated Readability Index} = 4.71\left(\frac{\text{characters}}{\text{words}}\right) + 0.5\left(\frac{\text{words}}{\text{sentences}}\right) - 21.43 \tag{4.4}$$

The `coleman_liau_index` function (Equation 4.5) is a weighted sum of the number of letters (without punctuation) in 100 words and average number of sentences per 100 words. According to this metric, given 100 words of text, the higher the characters, higher the complexity; however, the complexity is greatly reduced by increasing the number of sentences in the same span. This means that when a sentence is split, the grade level required to read the sentence drops. For example, to reduce the grade level of the text by approximately 1 point, we must split the sentences in the text such that there are 3 smaller additional sentences in the same span. This metric is affected by sentence length to a larger extent compared to other metrics.

$$\text{Coleman Liau Index} = 0.0588 \left(\frac{\text{letters}}{100 \text{ words}} \right) - 0.296 \left(\frac{\text{sentences}}{100 \text{ words}} \right) - 15.8 \tag{4.5}$$

The `linsear_write_formula` function (Equation 4.6, Equation 4.7, Equation 4.8) assigns 1 point to every word in the text and 2 extra points to words with 3 or more syllables. The average points per sentence is used to determine the complexity of the text. The higher the number of points, higher is the complexity. An interesting thing to note is that this metric does not penalize very long words any more than medium sized words.

$$\text{points} = \begin{cases} 1 & \text{words with two or less syllables} \\ 3 & \text{words with three or more syllables} \end{cases} \tag{4.6}$$

$$r_{\text{provisional}} = \frac{\sum points}{\text{number of sentences}} \tag{4.7}$$

$$\text{Linsear Write} = \begin{cases} r_{provisional} \ / \ 2 & r_{\text{provisional}} > 20 \\ (r_{provisional} \ / \ 2) - 1 & r_{\text{provisional}} \leq 20 \end{cases} \tag{4.8}$$

The `dale_chall_readability_score` function (Equation 4.9) is very similar to the `gunning_fog` function Equation 4.2 in terms of the parameters used. One note-able difference seems to be that this score attributes complexity of text to the percentage of difficult words more than `gunning_fog` does.

$$\text{Dale-Chall Readability} = 0.1579 \left(\frac{\text{difficult words}}{\text{words}} \times 100 \right) + 0.0496 \left(\frac{\text{words}}{\text{sentences}} \right) \tag{4.9}$$

The `flesch_reading_ease` function (Equation 4.10 is a flipped version of the `flesch_kincaid_grade` function which uses the same parameters to convey how easy the text is to read, instead of how complex it is to read. This means that a higher score indicates simpler text.

$$\text{Flesch Reading Ease} = 206.835 - 1.015 \left(\frac{\text{total words}}{\text{total sentences}} \right) - 84.6 \left(\frac{\text{total syllables}}{\text{total words}} \right) \tag{4.10}$$

Finally, the `text_standard` function (Equation 4.9) is a consensus function that looks at all the scores calculated by the functions above to provide the consensus, or the most common predicted grade level of the text.

We apply the all the above functions for each of sample of the aligned corpus (around 1000 samples) to create the box-plot in Table 4.2. The outliers are left out of the plot for all scores except the SMOG Index.

4.2.2. Interpretation

Taking a first look at Table 4.2 tells us that both E-Law as well as EasyLaw texts are fairly hard to read. With the exception of `smog_index`, all other scores tell us that, on an average, one needs to have a grade level of 15 or more, which is a college level education, in order to understand either of these texts. In fact, if we look at the Readability Consensus value, which is the most common predicted grade level of the text, we see that the median grade level required to be able to read and understand an E-Law text is less than the grade level required to read and understand an EasyLaw text. However, the mean Readability Consensus tells us that the grade level required to be able to read and understand an E-Law text is higher than the grade level required to read and understand an EasyLaw text. This implies that the E-Law texts have a higher variety of different grade level texts, with a large portion the texts having a lower readability score but a small portion of them which are outliers having a disproportionately high grade level. Already, just by analysing this metric, we can see that, if we assume that the EasyLaw service does indeed provide "Simple and Easy Legal Info", then these grade based readability metrics do not seem to be a good representation of "simple" legal text.

Most of the grade based metrics show a similar median grade level for the EasyLaw and E-Law texts which implies that both the texts require similar levels of education to read it. An exception is the Coleman Liau Index which appears to indicate higher median grade level for EasyLaw text - this could be explained by either a high words per sentences, that is, longer sentences or a high letter per word or longer word length. We can guess that longer sentences is not the cause since other metrics such as Flesch Kincaid grade, Gunning fog index, Automated Readability Index, Dale-Chall Readability and Flesch Reading Ease depend on the this metric but do not exhibit the same pattern. This means that EasyLaw exhibits a higher average letter count than E-Law texts. However, we do not observe a higher character per word count (includes punctuation) - this would have been the case if any of the metrics based on characters per word were higher for EasyLaw. This tells us, that the E-Law texts include a lot more punctuation on average than EasyLaw.

The SMOG Index also tells us that most of the EasyLaw texts (which as we may recall from chapter 3 contains a standalone piece of information) contain less than 3 sentences since the median of EasyLaw SMOG Index seems to be 0.

The takeaway here seems to be that length-based readability metrics as a whole do not give us a clear picture on the simplicity of legal texts. However, we can still glean some interesting properties of the text by looking at these metrics.

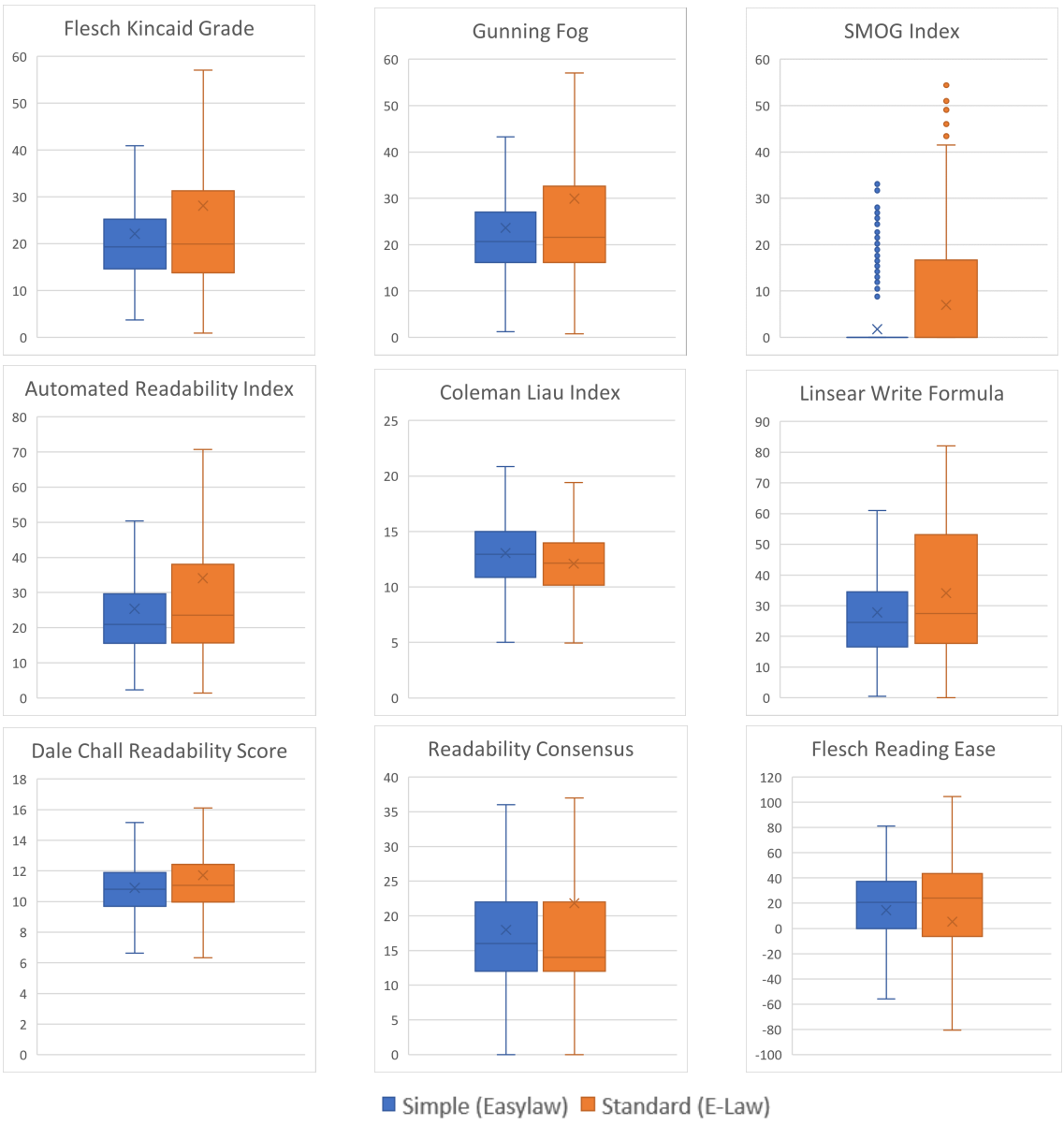

Table 4.2.: Grade based readability scores of EasyLaw vs. E-Law texts.

4.3. Structural features

In this section we explore the structural features of the EasyLaw and E-Law texts. An interesting manner of identifying the complexity of these texts would be to identify if the text satisfies the simple text writing guidelines mentioned in subsection 2.1.1. Since there is no direct way of doing this in an automated manner, we attempt to analyse the parse trees of the text to see if we are able to spot any differences in the frequently occurring parse structures of simple legal text. We refer to some linguistic studies and guidelines on plain English writing in general as well as for legal texts [35, 36][5]. These articles mention various structural characteristics of simple or plain English and we would like to understand if the EasyLaw texts reflect these guidelines. This is an important aspect to understand since it is not always feasible to adopt all these guidelines for the legal context. For practical purposes, not all of these guidelines can be quantitatively measured. For example, one guideline states, "Express only one idea per paragraph." However, it is very hard to measure or count "ideas". But there are other features that we can quantitatively measure such as usage of passives, references to other parts of the text etc.

4.3.1. Grammatical structure

Kauchak et al. [35] discuss that an accurate representation of the grammatical structure of a sentence is its parse tree expansion at a depth of 3. A sample 3rd level parse can be seen in Figure 4.1. The argument for this, is that at level 2, there is only one parse expansion and this leads to a very generic representation of a sentence. For example, most sentences have a Noun Phrase (NP) and a Verb Phrase (VP) and hence the parse structure at depth of two would just be "S [NP VP]" for most sentences. However, the parse tree expansion at a depth of 4 is too specific and may not allow us compute the frequencies of the grammatical structures since there maybe too few sentences that fall under each 4th level parse structure. So whenever possible, we use the 3rd level parse. We compute the 3rd level parse for EasyLaw and E-Law texts and the top 10 grammar structure frequencies can be seen in Table 4.3

From Table 4.3, we see that the top frequently occurring grammar structure is the same in both E-Law and EasyLaw. These are sentences like "An individual under influence may not drive." [(An individual [NP]) (under influence [PP]) (may [MD]) (not drive [VP]).] which have a Noun Phrase (NP) with a prepositional phrase (PP) and a Verb Phrase (VP) that starts with a modal verb (MD). It is interesting to note that the second most frequent structure in the standard E-Law texts is just a Noun Phrase (NP), i.e., not a complete sentence. However, the second most frequent EasyLaw structure is a complete sentence like "An individual who is under influence may not drive." [(An individual [NP]) (who is under influence [SBAR]) (may [MD]) (not drive [VP]).] - here the difference is that the sentence contains a subordinate clause (SBAR) instead of a prepositional phrase (PP). As we see in Figure 4.2, simple legal text seems to have complete sentences more often than standard Acts. This seems to indicate that complete sentences are simpler to understand. The work on improving readability with

[5] `https://www.llrx.com/2002/11/features-plain-language-in-law/` accessed on 08-10-2021.

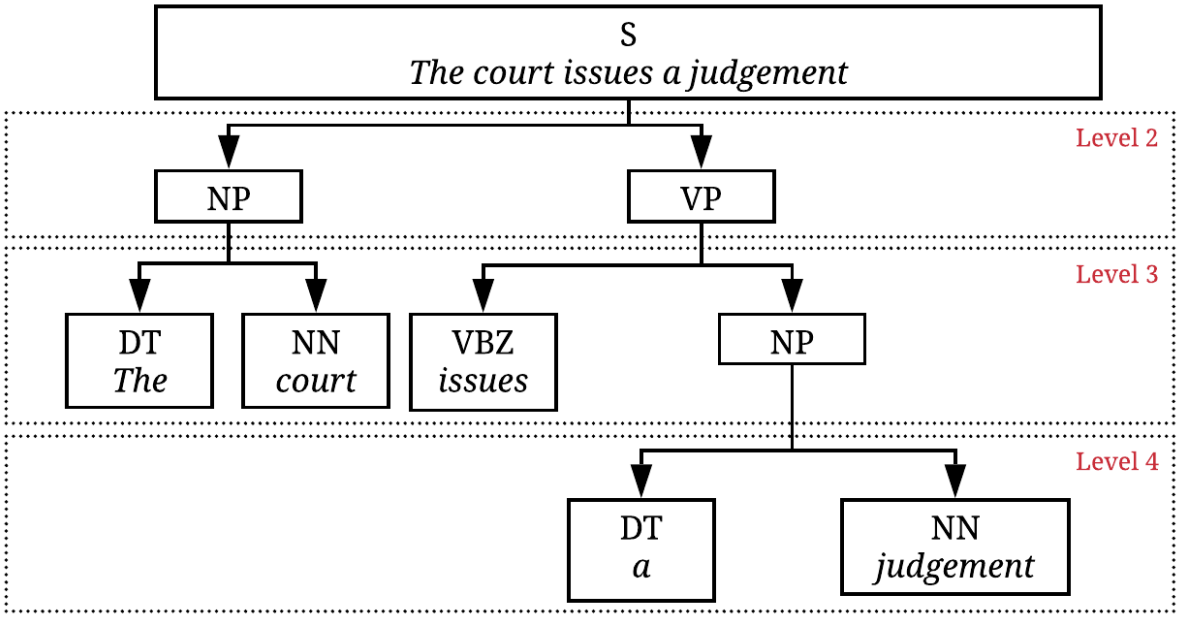

Figure 4.1.: Parse tree obtained via Constituency Parsing

Table 4.3.: Top 10 frequently occurring grammatical structures in EasyLaw and ELaw texts

Extracted Act (EasyLaw)	Actual Act (ELaw)	Rank
S [NP [NP PP] VP [MD VP]]'	S [NP [NP PP] VP [MD VP]]'	1
NP [NP [DT NN] SBAR [WHNP S]]'	S [NP [NP SBAR] VP [MD VP]]'	2
S [VBN PP [IN NP]]'	NP [NP PP [IN NP]]'	3
LST [NN]'	S [NP [NP VP] VP [MD VP]]'	4
S [NP [NP NP] VP [VBZ NP]]'	S [NP [DT NN] VP [MD VP]]'	5
S [VP [VBN PP]]'	FRAG [IN NP [NP SBAR]]'	6
NP [LS]'	S [SBAR [IN S] NP [DT NN] VP [MD VP]]'	7
S [VP [VBN SBAR]]'	S [NP VP [MD VP]]'	8
S [NP [NP SBAR] VP [MD VP]]'	NP [NP [DT NN] SBAR [WHNP S]]'	9
NP [NP [NN NNP NN CD] NP [NNP CD CD]]'	NP [NP [DT NN] PP [IN NP]]'	10

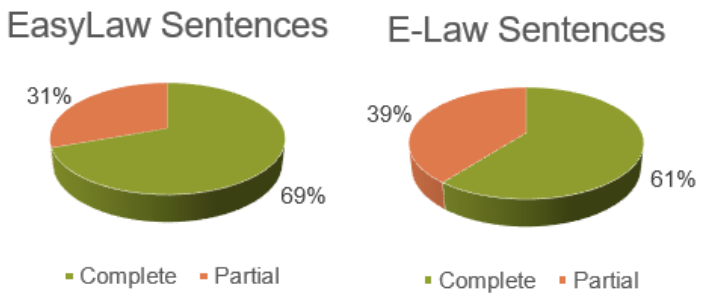

Figure 4.2.: Percentage of complete vs. partial sentences

syntactic cues [36] seems to suggest that usage of modal verbs such as can, should, may etc. improves the reader's ability to analyze and understand the structure of the sentence better. We look at the presence of modal verbs in the top 10 frequent grammar structures of the EasyLaw and E-Law texts. We see, in Figure 4.3, that the simple EasyLaw texts use modal

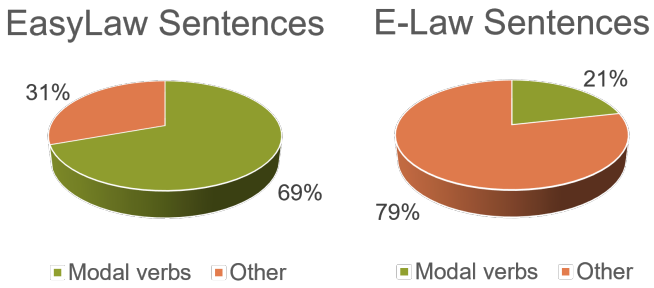

Figure 4.3.: Percentage of sentences with and without modal verbs.

verbs to form sentences more than 3 times as often as E-Law sentences do.

If we want to analyse the presence of passive voice, 3rd level parse structures are insufficient. The nature of the verb is seen only in the 4th level. For example, in Figure 4.4 we see that the verb indicative of passive which is the past participle (VBN) - for example, 'evicted' - occurs in the fourth level of the parse tree. Another indicator of a passive sentence is the verb phrase (VP) of the form [MD VP [VB VP]] which is a modal verb followed by a simple verb and a verb phrase - for example, "may be exempted". These are two common patterns we observe in the parse trees of passive sentences. We look at the top 20 most common frequencies at this level of parse and see from Figure 4.5 that E-Law text contains passive sentences almost twice as often as the EasyLaw text.

4.3.2. References

Another interesting statistic to look at is the frequency with which the EasyLaw and E-Law text refers to other parts of the text. We do this by examining the texts and looking for patterns of references. Some phrases used to refer to other parts of the text are "provisions of

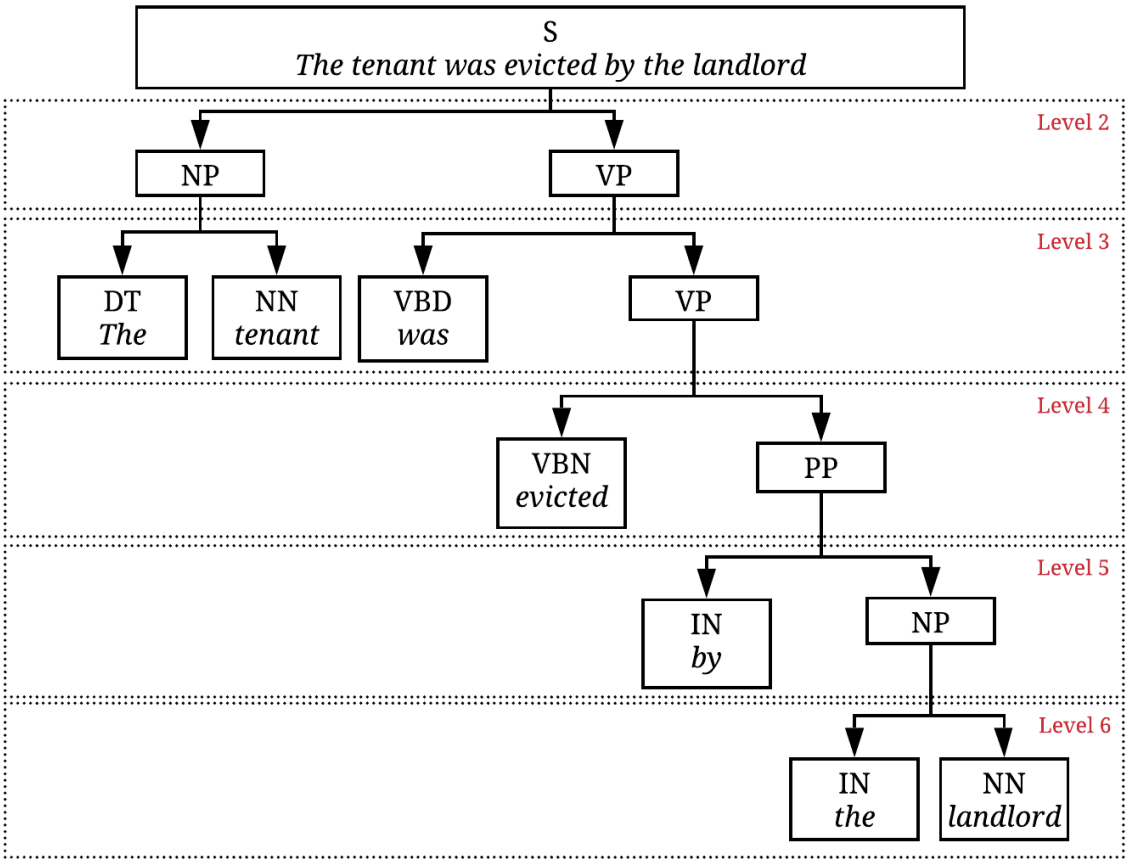

Figure 4.4.: Parse tree of a passive sentence.

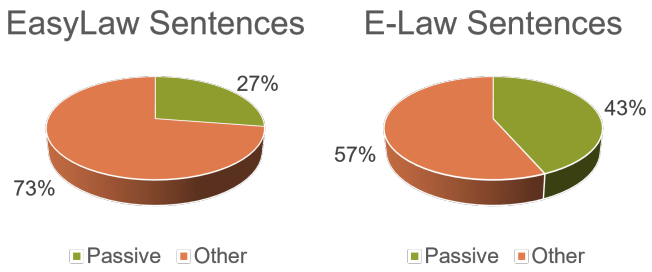

Figure 4.5.: Percentage of passive vs active sentence constructions.

Article", “violation of Article", “violated Article", “under Article". Counting the frequency with which these references occur, shows us that these references occur much more frequently in case of the E-Law texts.

Already, from Table 4.4, we see that we are 6 times more likely to see references to other parts of the text in the E-Law texts vs. the simpler EasyLaw texts. Moreover, each of these references in the E-Law texts refer to multiple parts of the text, as seen in the example in Figure 4.6, which compounds the difficulty in understanding the text.

Table 4.4.: How often do sentences refer to other parts of the text?

	Extracted Act (EasyLaw)	Actual Act (E-Law)
References	339	96539
Total lines	15k	800k
Percentage of references	2%	12%

The driver of any motor vehicle or tram or the rider of any horse who has violated Article 5, 13 (1) through (3) (in cases falling under Article 13 (3), excluding a person who drives a motor vehicle intentionally violating it on an expressway, an exclusive road for motor vehicles, or a road with median strip), or (5), 14 (2), (3), or (5), 15 (3) (including where the provisions are applied mutatis mutandis in Article 61 (2)), 15-2 (3), 16 (2), 17 (3), 18, 19 (1), (3), or (4), 21 (1), (3), or (4), 24, 25 through 28, 32, 33, 34-3, 37 (excluding paragraph (1) 2), 38 (1), 39 (1), (3), (4), and (5), 48 (1), 49 (excluding a person who drives a motor vehicle or tram in violation of paragraph (1) 1 or 3 of the same Article and a person who drives a motor vehicle with a device installed on which to interfere with functions of traffic regulation equipment, of the offenses referred to in subparagraph 4 of the same paragraph), 50 (5) through (7), 51, 53 (1) or (2) (excluding a driver who fails to make children or infants fasten their seat belts), 62, or 73 (2) (excluding subparagraph 1 of the same paragraph)

Figure 4.6.: Example of a law which references multiple other articles and acts - Article 156 of the Road Traffic Act [6]

4.4. Discussion

In this section, we look at different ways of measuring complexity of text and apply these measures to compute the complexity of the simple and standard laws collected in chapter 3. We see that EasyLaw texts have lower count-based complexity scores. But it is hard to interpret whether EasyLaw texts have lower count scores because they contain less information or

[6] `https://elaw.klri.re.kr/eng_service/lawView.do?hseq=50713&lang=ENG` accessed on 08-10-2021

because they actually have simpler text. However, if we look at the length-based complexity scores which are, in most cases, linear combinations of count-based scores, we see that the EasyLaw and E-Law have comparable scores in most cases. This means that the difference in count based scores is due to extra information present in the E-Law texts. This also correlates with our observations in section 3.4. Hence, traditional count-based or length-based readability scores do not seem to offer much information regarding the true complexity of legal texts. This indicates that we must look for new or different metrics to measure the complexity of legal texts.

We also look at some structure based metrics by trying to measure adherence of the simple and standard texts to some of the "Plain Language" guidelines in literature. These measures show a promising ability to distinguish between simple and standard legal text to varying degrees. However, these metrics are all manually calculated based on observations on a small sample of the data. If we could cross-check these findings by employing most sophisticated methods on a larger number of data samples, we would be able endorse these metrics with a higher confidence. The number of cross-references seems to be a clear distinguishing factor between complex and simple texts and we could expand this metric by also calculating how many references are present in each sample rather than just looking at how many samples contain references. This also provides a way to measure the degree of referencing and might be an even better indicator of complexity. In this thesis we only look at some sample referencing patterns but another way to expand this metric would be to mine more reference patterns - this would make the metric more robust.

5. Text Simplification Models on Legal Text

In this section, we explore the state-of-the-art text simplification models for two kinds of text simplification. We apply these models on the collected standard legal text (E-Law texts) and produce simplified versions of the E-Law text. We then compare the performance of the models on these texts and discuss their successes and pitfalls and their suitability to be applied on legal texts.

5.1. LS-Bert

LS-Bert or Lexical Simplification BERT [30] Figure 5.1 is a lexical simplification model. This means that the model attempts to recognize complex words in the input text and replace them with simpler words without changing the structure of the text.

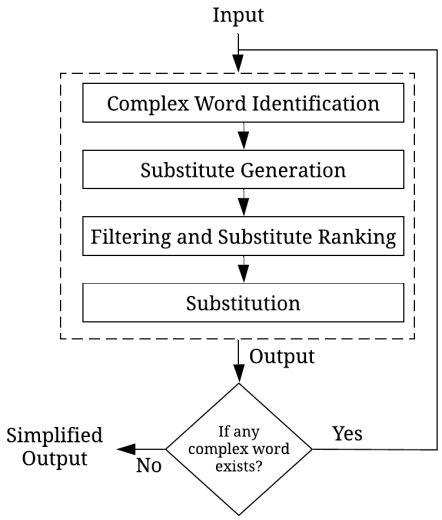

Figure 5.1.: Illustration of the LS-BERT model by Qiang et al. [30]. The model recursively tries to simplify any complex words it detects.

This model employs a complexity classifier on each word of the sentence. It picks the most complex word and generates possible replacements for the word by masking it and feeding the masked sentence to BERT which is a masked language model. (A Masked Language Model is capable of filling in the masked words with the most likely replacements.) BERT then generates suitable replacements which are ranked based on whether the generated candidate is a paraphrase of the original word, its language model score, how common the

word is and its semantic similarity to the original word. The best replacement is then chosen and the model then moves to the next most complex word in the sentence and performs the entire sequence again. This is repeated until no more complex words are left in the sentence.

5.1.1. Generating simplifications

We use the LS-Bert code and model published on GitHub[1] to generate the simplifications of E-Law texts. Due to the sequential word-by-word application of the model, as well as the use of multiple models (complexity detection model, BERT, paraphrase model), this model is quite slow to apply. The model took, on an average, approximately 10 minutes to simplify 1 line of E-Law text. We generated simplified sentences of around 1500 lines of text from E-Law from our aligned corpus using the LS-Bert model.

5.2. MUSS+ACCESS

The MUSS+ACCESS model [2] is general paraphrasing model which means that it attempts to simplify text by performing a variety of transformations such as sentence splitting, voice change, tense change, lexical simplification etc. MUSS stands for Multilingual Unsupervised Sentence Simplification. Here, the training data is mined in an unsupervised manner by labelling sentences with close language model scores as paraphrases. Once the paraphrase data is mined, it is used to train the ACCESS model [37]. ACCESS stands for AudienCe-CEntric Sentence Simplification and this model is trained by appending certain signals to the source text. Any two paraphrases are chosen as source and target and signals relative to these two texts are attached to the source text. For example, the signal could be the amount of compression seen in the target text with respect to the source text in terms of number of characters. In Table 5.1, we see that the number of characters in target text is 0.3 times the number of characters in the source, and hence the token $< NbChars_{0.3} >$ is attached to the source. At decode time, this token is then used to determine how short we want our target text to be.

5.2.1. Generating simplifications

We use the MUSS+ACCESS code and model published on GitHub[2] to generate our simplifications. Using the MUSS + ACCESS model, we generated simplified sentences of the same 1000 E-Law samples from our aligned corpus as we used for the LS-Bert model.

5.3. Quantitative comparison

In this section we take the simplifications obtained by applying the LS-Bert and MUSS+ACCESS models on the standard laws (E-Law texts) and compute their text complexity scores as dis-

[1] `https://github.com/qiang2100/BERT-LS` accessed on 08-10-2021
[2] `https://github.com/facebookresearch/muss` accessed on 08-10-2021

Source Text	*<NbChars 0.25>* Some trails are designated as nature trails , and are used by people learning about the natural world .
Target Text	Some trails are used by people .

Table 5.1.: One of the examples shown by Martin et al. illustrating the use of control tokens to control the target text obtained from the model. [37]

cussed in section 4.1 and section 4.2. We visualise these scores along with those of the original E-Law texts to understand how these simplification models affect these scores and to understand any insights we may get regarding the type of transformation performed by the models.

We first examine the count-based complexity metrics in Table 5.2. Looking at the syllable count and word count (lexicon), we clearly see that MUSS simplified texts seem smaller in both aspects - which implies that the paraphrase transformations performed by MUSS + ACCESS reduces the number of words in the sentence while simplifying, which in turn also reduces the number of syllables in the text - both implying shorter sentence lengths. Neither model seems to alter the number of sentences to a great extent. The Difficult Words measure shows a different trend compared to the other metrics we've seen so far. It shows that LS-Bert is able to reduce the number of complex words in the text to a higher extent compared to MUSS + ACCESS. This is not surprising, given that LS-Bert performs complex word identification as a separate step and then attempts to replace these words.

Next, we examine the length-based complexity metrics in Table 5.3. A first look at the box-plot of metrics shows us that, most of the time, MUSS + ACCESS model is able to simplify text better according to the length-based metrics. This is not surprising since the MUSS + ACCESS model actually selects the best model based on Flesch Kincaid Grade Level score, among other scores, on the validation set, whereas the LS-Bert does not change the number of words per sentence which contributes a major weight in most of these measures. Looking at the two metrics that do not follow the same trend, namely, SMOG Index and Coleman Liau Index, we see that both depend heavily on length of the word, i.e., SMOG Index on number of polysyllable words per sentence and Coleman Liau on the number of letters per word. Since LS-Bert performs lexical simplification, i.e., simplification on a word level, this directly affects the complexity of the word and leads to replacement of bigger words with a easier, often smaller word. The fact that these complexity scores are lower indicates that LS-Bert indeed performs more simplification on a word level than MUSS + ACCESS.

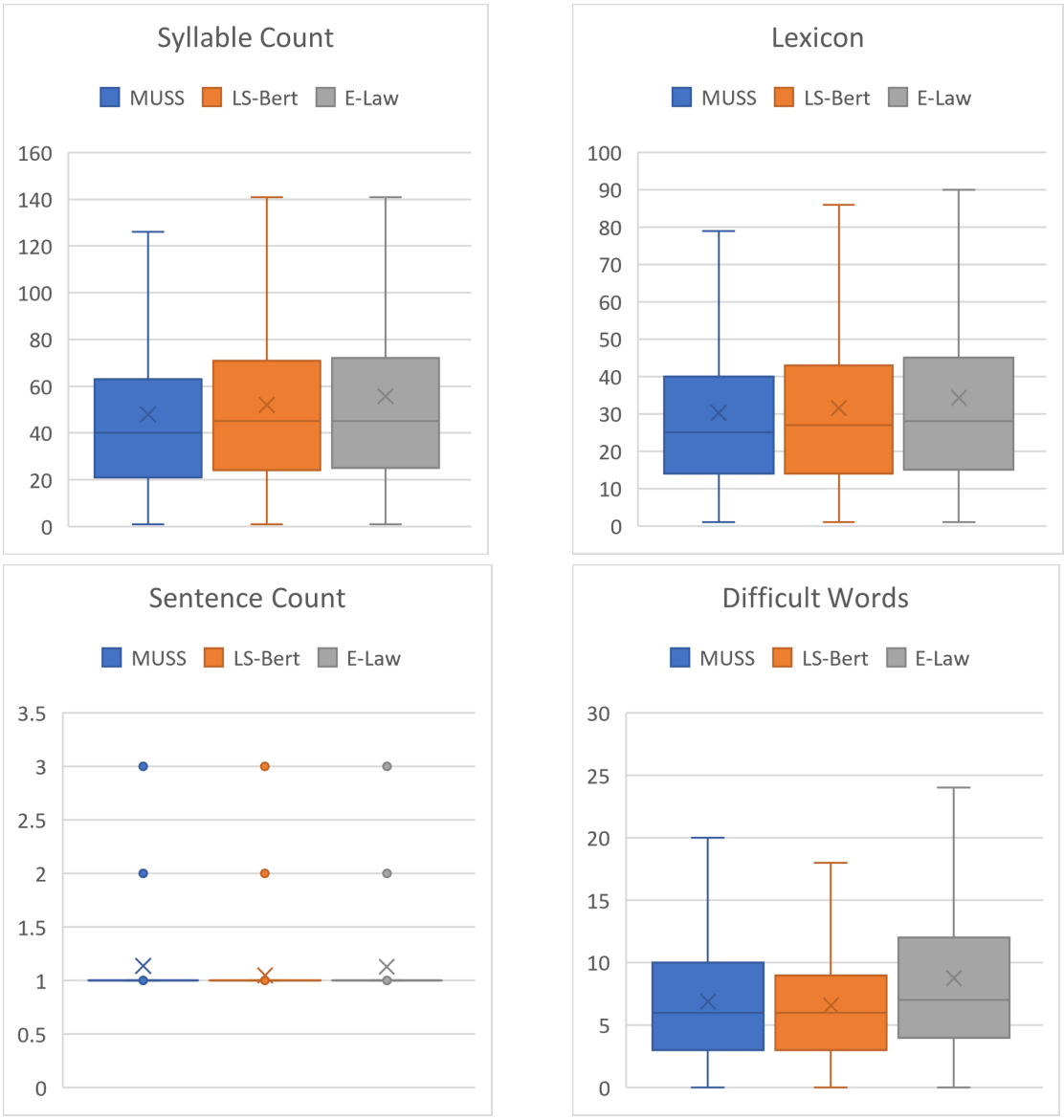

Table 5.2.: Count based readability scores of EasyLaw vs E-Law texts.

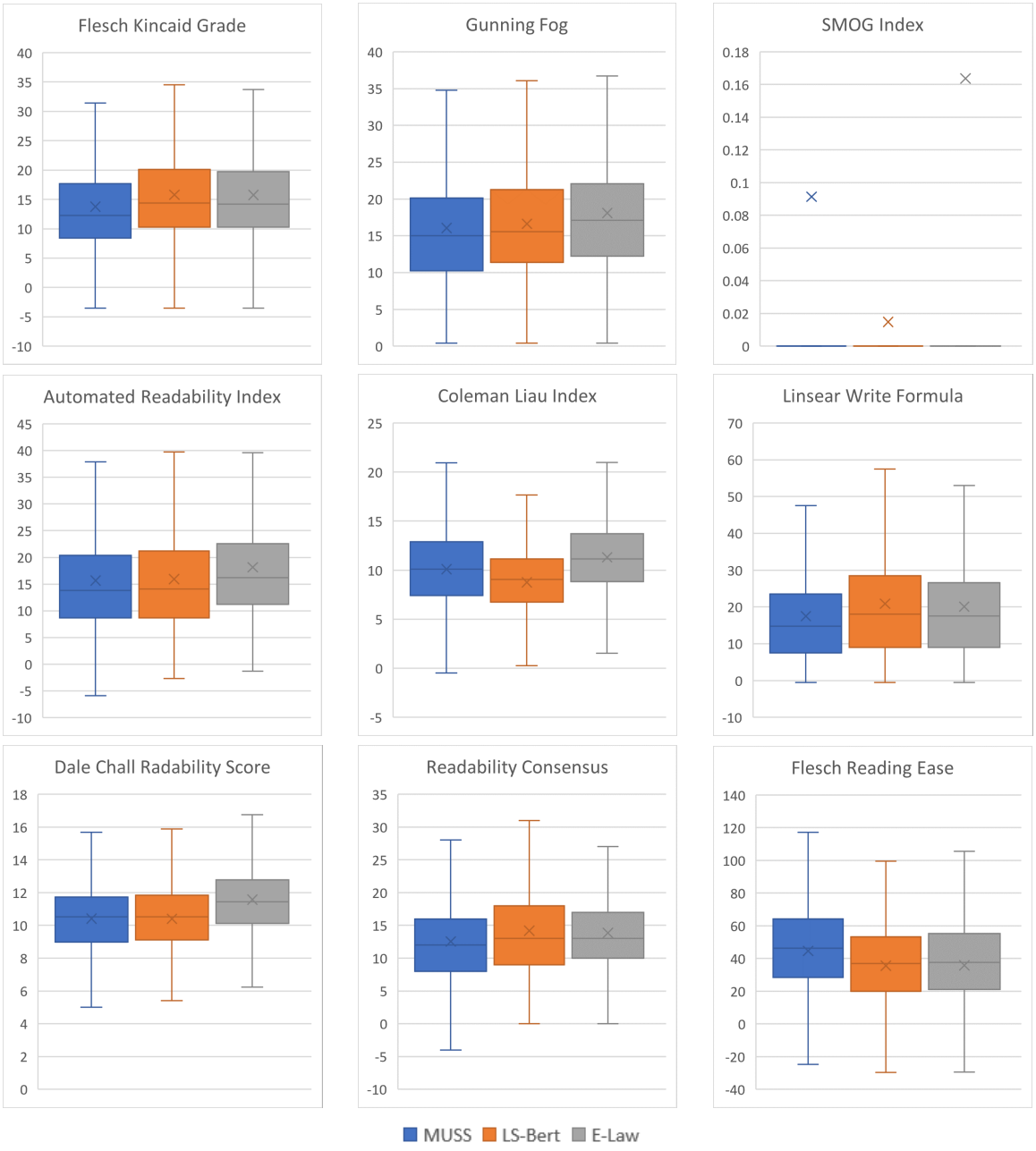

Table 5.3.: Grade based readability scores

5.4. Qualitative comparison

In this section, we look at some example simplifications performed by the models on various articles of the Civil Act from E-Law[3] and spot the successes and failures of each of the models.

In Table 5.4, we see that the MUSS+ACCESS model is able to simplify some of the complex E-Law phrases. For example, “shall obtain the " has been transformed to “must get the" which is a successful simplification. Another good simplification is the transformation of the phrase “exercise the right of consent" into “give consent". Overall this model seems to have made a successful simplification of this statute.

The LS-BERT model has transformed the word ‘minor’ to the ‘child’ and ‘marry’ to ‘meet’. Both these changes lead to loss of meaning. In this case, LS-BERT has failed to produce a good simplification.

Text Source	**Example**
E-Law	A minor shall obtain the consent of both parents in order to marry. If one parent is unable to exercise the right of consent, the minor shall obtain the consent of the other parent, and if neither parent is able to exercise the right of consent, the minor shall obtain the consent of his or her guardian of minor.
EasyLaw	In case a minor or an adult ward marries, he/she shall go through the following procedures for consent. A minor (under the age of 19) shall obtain the consent of both parents in order to marry. If one parent is unable to exercise the right of consent, the minor shall obtain the consent of the other parent, and if neither parent is able to exercise the right of consent, the minor shall obtain the consent of his/her guardian of minor.
MUSS+ACCESS	A minor must get the consent of both parents in order to marry. If one parent is not able to give consent, the minor must get the consent of the other parent. If neither parent is able to give consent, the minor must get the consent of the minor’s guardian.
LS-Bert	A child will receive the permission of both parents in order to meet. If one parent is unable to use the right of permission, the child must get the permission of the other parent, and if neither parent is able to use the right of permission, the child will get the permission of his or her father of major.

Table 5.4.: Simplification of Article 5 of the Civil Act by LS-Bert and MUSS.

[3] `https://elaw.klri.re.kr/eng_service/lawView.do?hseq=29453&lang=ENG` accessed on 08-10-2021.

In Table 5.5, we see that MUSS + ACCESS also falters in terms of retaining the complete information because it has dropped the information “harvest season” and just talks about a generic “season”. LS-Bert does retain some information that conveys that the “season” refers to something related to agriculture by replacing “harvest” with “crop”. But here the meaning of the text is changed since “crop year” refers to a different time period than “harvest season” - so this simplification would result in misinformation.

Text Source	**Example**
E-Law	Rent shall be paid at the end of each month in the case of a movable, a building, or a building site, and at the end of each year in the case of any other land: Provided, That in the case of a leased object which has a harvest season, rent shall be paid, without delay, upon the close of such a season.
EasyLaw	If the lessee and the lessor have not determined the timing to pay rent, rent shall be paid at the end of each month
MUSS+ACCESS	Rent shall be paid at the end of each month in the case of a movable object, a building or a building site, and at the end of each year in the case of any other land. The rent shall be paid, without delay, upon the close of such season in the case of a leased object.
LS-Bert	it will be paid at the end of each month in the case of a fixed , a building , or a building site , and at the end of each year in the case of any other land : Provided , That in the case of a used subject which has a crop year , rent will be paid , without cause , upon the close of such a season .

Table 5.5.: Simplification of Article 633 of the Civil Act by LS-Bert and MUSS.

In Table 5.6, MUSS + ACCESS performs no major simplification but reorders some of the phrases in the sentence - hence there is no loss of information but also no reduction in complexity. However, LS-Bert has again modified words that carry special meaning; such as converting “lineal ascendants” to “female parents” - but “lineal ascendants” in fact refers to parents, grandparents and so on, of any gender.

Text Source	Example
E-Law	The lineal ascendants of a parent who does not foster children may request the Family Court to grant visitation with the children, where the parent is unable to visit the children due to death or any other extenuating circumstance such as illness and residency abroad. In such cases, the Family Court shall take into consideration circumstances such as the intent of the children, the relationship between the requesting person and the children, and the reasons for the request.
EasyLaw	In addition, a parent who does not foster children and his/her children shall have the visitation right. If it is required for the welfare of children, the Family Court may, upon a request of the party or ex officio, restrict or exclude such visitation right.
MUSS+ACCESS	The lineal ascendants of a parent who does not foster children may request the Family Court to grant visitation with the children, where the parent is unable to visit the children due to illness, residency abroad, or other extenuating circumstance. In such cases, the Family Court shall take into consideration circumstances such as the intent of the children, the relationship between the requesting person and the children, and the reasons for the request.
LS-Bert	The female parents of a parent who does not raise children may request the Family Court to grant custody with the children , where the parent is unable to visit the children due to death or any other such situation such as sickness and detention abroad . In such cases , the Family Court will take into account circumstances such as the purpose of the children , the relationship between the receiving person and the children , and the reasons for the demand .

Table 5.6.: Simplification of Article 837 of the Civil Act by LS-Bert and MUSS.

5.5. Discussion

In this chapter, we have seen how well the standard versions of LS-Bert and MUSS + ACCESS perform on legal text. As we can see, MUSS seems to perform less risky transformations which are more likely to retain original information, but this also means that the degree of simplification is less. We can attempt to overcome this problem by experimenting with the control tokens in the ACCESS model. For example, the control token 'WordRank' is used by the ACCESS model to indicate how much the complexity of words in the simplified sentence should be reduced. The WordRank metric is calculated based on how common a word is. Martin et al. [2] have performed an extensive study on various control tokens and determined that WordRank of 0.75 provides the best simplification. However, experimenting with this token for legal domain texts may let us use a smaller WordRank which gives simplified legal texts with simpler words. This is a promising direction since it is imaginable that the legal domain has more complex sentences than a random sentence on the internet and hence a potential for higher reduction in WordRank.

LS-Bert's ability to simplify words, although hitherto unsuitable, hold promise if we are able to control the types of words it simplifies. For example, if we are able to detect all legal Named Entities [38], we could prevent the model from simplifying these. Another approach could be to allow the model to simplify only verbs, adverbs or adjectives as this would prevent the model from changing noun phrases that hold a special meaning in the legal domain. These kind of restrictions and checks could allow the model to simplify words without loss of meaning.

An adaptation of both these models specific to the legal domain, could already provide legal texts that are marginally simpler than what exists today.

6. Conclusion & Outlook

This chapter concludes the thesis by describing the key takeaways and limitations of the thesis. We also discuss the outlook and scope of any further work that can be performed by using this thesis as a starting point.

6.1. Thesis summary

In this thesis, we first created a corpus of around 717 structured simple law articles by scraping and processing the EasyLaw website. We then scraped and processed the E-Law website to create a corpus of around 2183 structured Acts written in the standard form. We then aligned the simple articles with the standard Acts. We were able to create a corpus of 922 aligned samples. While the method does choose the closest alignment based on the extracted metadata, the aligned simple and the standard text often do not contain exact same information. The standard law usually containing more information than its simple law counterpart.

We then use this data to calculate count-based complexity measures, length-based complexity measures, grammatical structure and the amount of cross-references in the text. We discover that, while count-based complexity measures indicate a higher complexity for the standard law, this may also be attributed to the extra information conveyed in them and hence these measures are not too reliable. Next we look at the length-based metrics and discover that these metrics do not always score the standard law as the more complex law - this leads us to conclude that these metrics are perhaps not ideal to measure simplicity/complexity in legal texts. However, they do help us understand some characteristics of the standard and simple legal texts. The standard texts, on an average, contain a lot more punctuation than simple legal texts, perhaps due to extensive referencing. On the other hand, simple texts seem to have less number of sentences per unit text. Looking at the grammatical structure indicates that, while both simple and standard legal text have similar mode sentence structures, standard legal text contains a larger percentage of partial sentences, sentences constructed in passive voice and a less frequent use of modal verbs, which are linguistic indicators of complex text. Similarly, we see that standard legal texts are 6 times more likely to use cross-referencing to refer to other parts of legal texts than their simple counterparts.

We also use the created corpus as input to current state-of-the-art simplification models to study how well these models simplify legal text. Examining the simplified texts quantitatively using length-based readability scores tells us that the models have indeed reduced the complexity of the legal text. However, when we investigate the simplification produced by these models, we see that the models often either retain or even increase the level of

complexity of these legal texts by changing its meaning or dropping information. We further see that lexical simplification models are more likely to cause information loss by changing the meaning of the text. Paraphrasing simplification models may drop information. This leads us to the conclusion that the current models need to be tuned, perhaps, based on complexity measures specific to legal text, to perform legal text simplifications and cannot be used directly on legal text.

6.2. Future outlook

We have discussed potential improvements and future work that could be done to build on and improve the corpus creation, complexity measurement as well as text simplification models in the individual chapters. Here, we discuss a more generic outlook and understand the motivation for future work the thesis provides. One main takeaway has been that the current text complexity measures are not representative of the complexity of legal text. We see this in the inability of the measures to differentiate between simple and standard legal texts as well as in the fact that models that are developed by using these metrics to detect complexity, are able to reduce complexity only in terms of these metrics but are unable to effectively simplify legal texts. This motivates us to look for more metrics that do represent the complexity of legal text like degree of cross-referencing or ways of measuring the grammatical structures that represent simple legal text and use these metrics as objectives while developing legal text simplification models.

Further, improving the created corpus and having it evaluated by domain experts would be a major step in creating a benchmark for measuring how well future text simplification models perform on legal texts. Such a data-set, even if small in size, could also be used to fine-tune large transformer models and teach them to perform legal text simplification, similar to what has been done for tasks such as summarization with dataset sizes varying from tens to few thousands with varying degrees of success [39, 40, 41, 42].

A. Code

A.1. Translating EasyLaw HTML into Python dictionaries

The function `parse_into_dict` converts an EasyLaw HTML file into a python dictionary. It first scrapes the relevant parts of the page, identifies the hierarchy of the legal text and stores it accordingly.

```
def keep_only_text(new_dict):
  if not len(new_dict['children']):
    return new_dict["text"]

  obj ={
      new_dict['text'] :[]
      }
  for child in new_dict['children']:
      obj[new_dict['text']].append(keep_only_text(child))
  return obj

def parse_into_dict(html_file):
  soup =BeautifulSoup(html_file, features="html.parser")
  dict ={
      "text" :soup.find('div', attrs={'class': re.compile("cnpClsTitle")}).text.strip(),
    "children": [],
    "pos1": -1,
    "pos2": -1
    }
  stack =[dict]

  for link in soup.findAll('div', attrs={'class': re.compile("plv([1-9])([a-z])")}):
    new_obj ={}
    p =re.findall("plv([1-9])([a-z])", link.attrs["class"][0])
    new_obj["pos1"] =ord(p[0][0]) -ord('1')
    new_obj["pos2"] =ord(p[0][1]) -ord('a')
    new_obj["text"] =link.text
    new_obj["children"] =[]

    while (stack[-1]['pos1'] <new_obj['pos1'] or
          (stack[-1]['pos1'] ==new_obj['pos1'] and
           stack[-1]['pos2'] <new_obj['pos2'])) ==False:
      stack.pop()

    stack[-1]["children"].append(new_obj)
    stack.append(new_obj)
  return keep_only_text(dict)
```

A.2. Translating E-Law HTML into Python dictionaries

The function `parse_elaw_into_dict` converts an E-Law HTML file into a python dictionary. It first scrapes the relevant parts of the page and extracts the divs that contain the E-Law act text. Then it uses the class tags to identify the hierarchy of the legal text in the div and stores it accordingly. The outermost level has the class tag 'JO', the next one has the tag 'HANG' and the innermost one has the tag 'HO'.

```
def traverse(node):
  d ={}
  if node.string:
    d["text"] =node.string.encode("ascii", "ignore").decode().strip()
  d["children"] =[]
  if node.has_attr("class"):
    d["class"] =node.attrs["class"][0]

  lasthang =None
  for child in node.children:
    if child.name =='div':# or child.name == 'span':
      child_node =traverse(child)
      if "class" in child_node and child_node["class"] =='hang':
        lasthang =child_node
        del child_node["class"]
        d["children"].append(child_node)
      elif "class" in child_node and child_node["class"] =='ho' and lasthang:
        del child_node["class"]
        if "children" not in lasthang:
          lasthang["children"] =[]
        lasthang["children"].append(child_node)
      else:
        d["children"].append(child_node)

  if not d["children"]:
    d["text"] =node.text.encode("ascii", "ignore").decode().strip()
    del d["children"]
  else:
    if "text" not in d:
      if "class" in d["children"][0]:
        if d["children"][0]["class"] in ['articletitle', 'lawname', 'addenda']:
          d["text"] =d["children"][0]["text"]
          del d["children"][0]
  return d
```

```
def to_tree(node):
  if "children" not in node:
    return node["text"]
  children =[]

  for n in node["children"]:
    children.append(to_tree(n))

  if "text" not in node:
    return {"Addendum":children}
  return {node["text"]:children}

def parse_elaw_into_dict(html_file):
  soup =BeautifulSoup(html_file, features="html.parser")
  tree =to_tree(traverse(soup.html.body.div))
  return tree
```

A.3. Regular Expressions for capturing Article information

`match_article` looks for the regex patterns stored in the patterns array, in the text and returns all matches.

```
def match_article(text):
    patterns =[
    # Pattern to capture references starting with Article...
    re.compile(r'(Article [0-9]*)[-.(]+[0-9]+)?([ ).-]+[0-9]+)?([ .-]*[A-Z])? of the((\s((of)|(
                                                    the)|(on)|(and)|([A-Z][a-z-]*)))+)((?= and
                                                    Article)|(?=.)|(?=))|(?=,))'),

    # Pattern to capture references starting with Subparagraph...
    re.compile(r'Proviso to Subparagraph (?P<section>[0-9]*) of Article (?P<article>[0-9]*) of
                                                    (?P<act>[A-Za-z0-9.\s-]*)((?=))|(?=,)|(?=
                                                    and Article)|(?=. ))'),
    ]

  matches =[]
  encoded_string =text.encode("ascii", "ignore")
  text =encoded_string.decode()
  for expr in patterns:
    for obj in expr.findall(text):
      matches.append({
        "Text":text,
        "Act":obj[4].strip(),
        "Article":obj[0],
        "Section":obj[1],
        "Subsection":obj[2],
        "Item": obj[3]
        })
    return matches
```

B. Figures

This section contains some figures. The Figure B.1 shows how a single EasyLaw excerpt has references to more than one E-Law Acts.

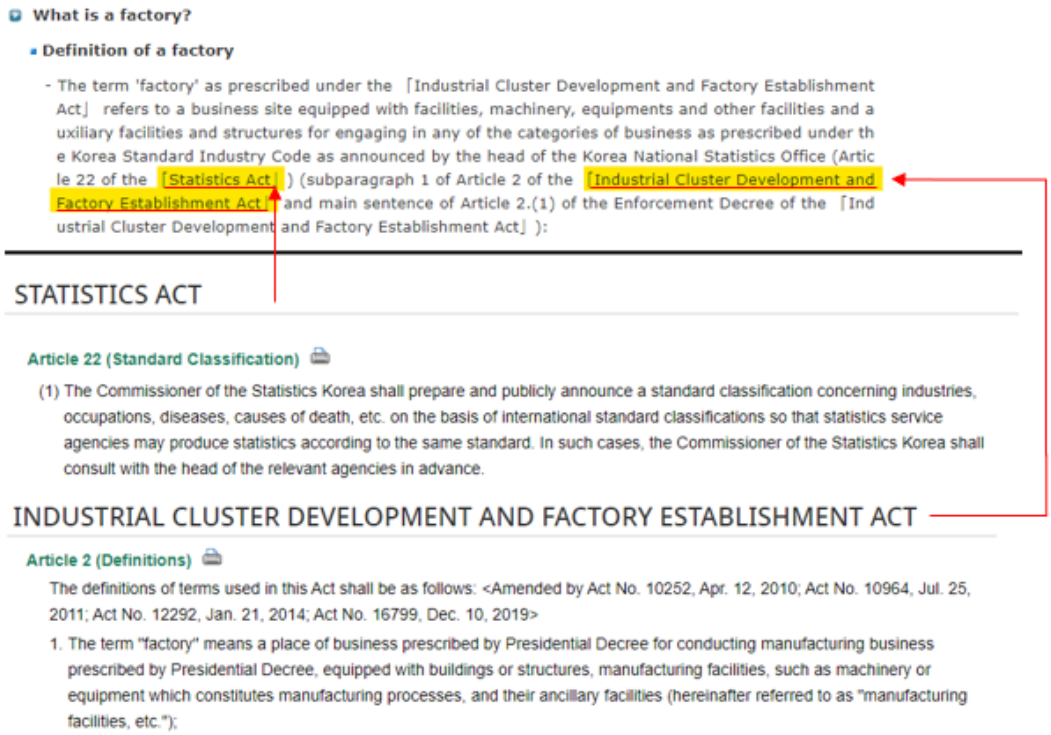

Figure B.1.: EasyLaw excerpt with corresponding E-Law Acts. Source: EasyLaw [1]and E-Law [2][3]websites. Annotations added to indicate references to E-Law from EasyLaw.

The Figure B.2 shows how the DOM tree of the web page shown in Figure B.3 is structured. It also shows an expanded div for further clarity on the structure.

The Figure B.3 shows a screenshot of the starting page of the Easy Law website. [4]

[1] `https://www.easylaw.go.kr/CSM/SubCnpclsCmd.laf?csmSeq=634&ccfNo=1&cciNo=1&cnpClsNo=1` accessed on 08-10-2021

[2] `https://elaw.klri.re.kr/eng_service/lawView.do?hseq=34658&lang=ENG` accessed on 08-10-2021

[3] `https://elaw.klri.re.kr/eng_mobile/viewer.do?hseq=42760&type=part&key=28&lang=ENG` accessed on 08-10-2021

[4] `https://www.easylaw.go.kr/CSM/SubMainCmd.laf?langCd=700101` accessed on 08-10-2021

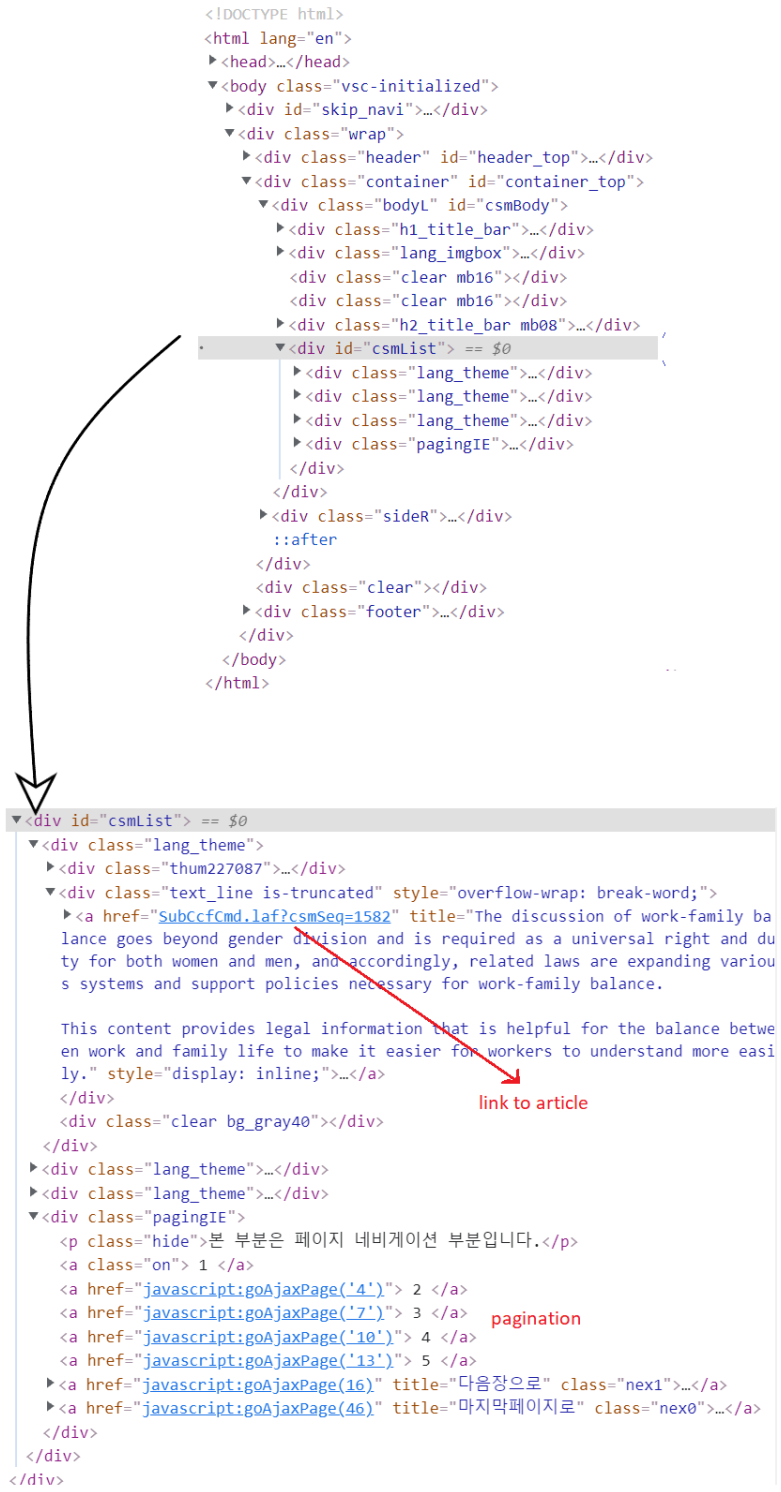

Figure B.2.: DOM tree of the webpage shown in Figure B.3

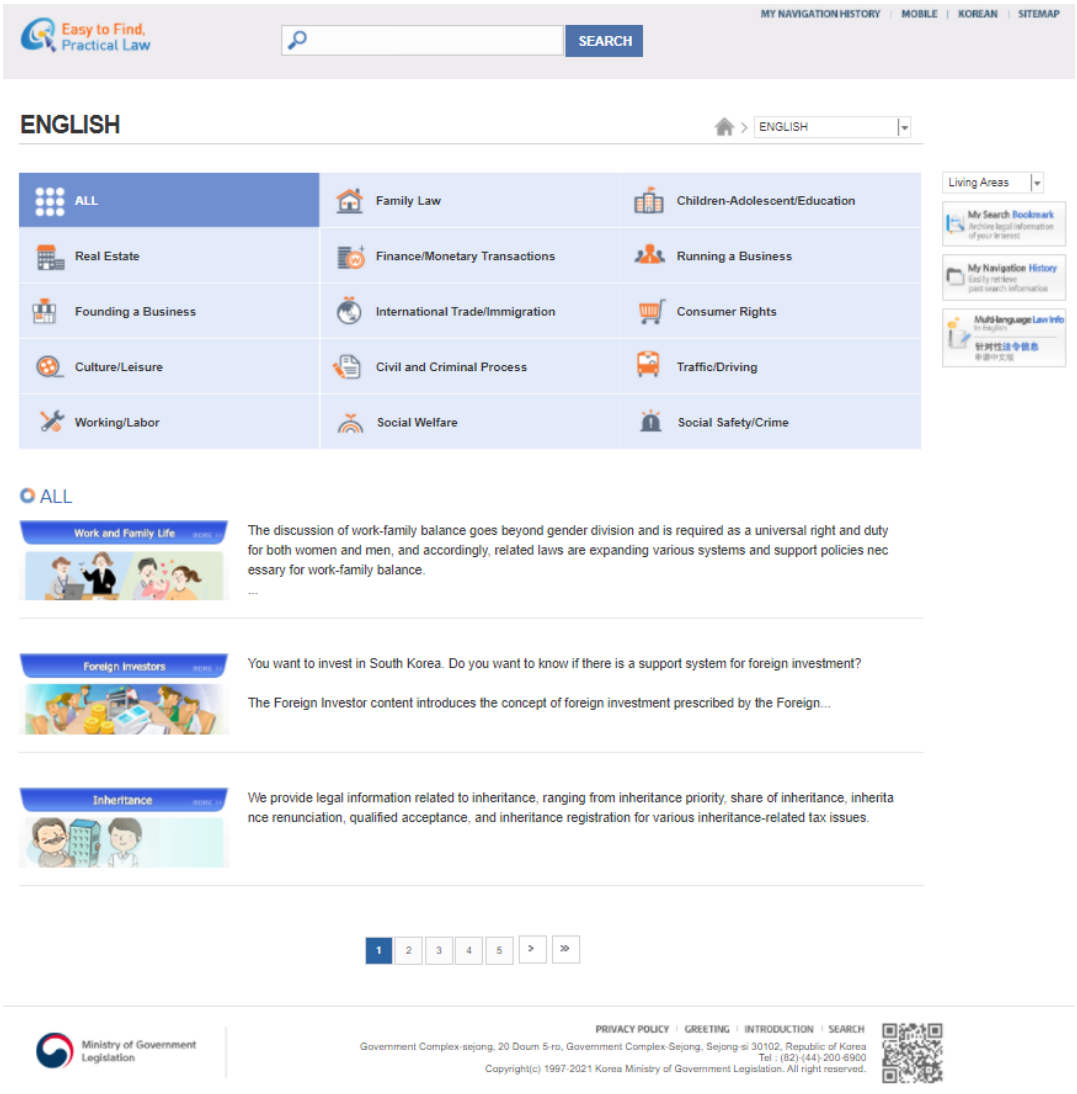

Figure B.3.: Starting page of the Easy Law website for scraping.

List of Figures

List of Tables

Bibliography

[1] D. I. Rubab, M. Y. Khan, and D. T. Asgher. "Transformation of Legal Texts into Simplified Accounts to Make the Justice Accessible". In: *Pakistan Social Sciences Review* 1 (Mar. 1), pp. 141–153.

[2] L. Martin, A. Fan, É. de la Clergerie, A. Bordes, and B. Sagot. "Multilingual Unsupervised Sentence Simplification". In: *CoRR* abs/2005.00352 (2020). arXiv: 2005.00352. URL: https://arxiv.org/abs/2005.00352.

[3] X. Zhang and M. Lapata. "Sentence Simplification with Deep Reinforcement Learning". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 584–594. DOI: 10.18653/v1/D17-1062. URL: https://aclanthology.org/D17-1062.

[4] J. Carroll, G. Minnen, D. Pearce, Y. Canning, S. Devlin, and J. Tait. "Simplifying Text for Language-Impaired Readers". In: *Ninth Conference of the European Chapter of the Association for Computational Linguistics*. Bergen, Norway: Association for Computational Linguistics, June 1999, pp. 269–270. URL: https://aclanthology.org/E99-1042.

[5] R. Bhagat and E. Hovy. "What is a paraphrase?" In: *Computational Linguistics* 39.3 (2013), pp. 463–472.

[6] C. Jiang, M. Maddela, W. Lan, Y. Zhong, and W. Xu. "Neural CRF Model for Sentence Alignment in Text Simplification". In: *CoRR* abs/2005.02324 (2020). arXiv: 2005.02324. URL: https://arxiv.org/abs/2005.02324.

[7] G. Paetzold, F. Alva-Manchego, and L. Specia. "Massalign: Alignment and annotation of comparable documents". In: *Proceedings of the IJCNLP 2017, System Demonstrations*. 2017, pp. 1–4.

[8] W. Xu, C. Callison-Burch, and C. Napoles. "Problems in Current Text Simplification Research: New Data Can Help". In: *Transactions of the Association for Computational Linguistics* 3 (2015), pp. 283–297. DOI: 10.1162/tacl_a_00139. URL: https://aclanthology.org/Q15-1021.

[9] Z. Zhu, D. Bernhard, and I. Gurevych. "A monolingual tree-based translation model for sentence simplification". In: *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. 2010, pp. 1353–1361.

[10] X. Zhang and M. Lapata. "Sentence Simplification with Deep Reinforcement Learning". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 584–594. DOI: 10.18653/v1/D17-1062. URL: https://aclanthology.org/D17-1062.

[11] W. Coster and D. Kauchak. "Simple English Wikipedia: A New Text Simplification Task". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 665–669. URL: https://aclanthology.org/P11-2117.

[12] W. Xu, C. Napoles, E. Pavlick, Q. Chen, and C. Callison-Burch. "Optimizing Statistical Machine Translation for Text Simplification". In: *Transactions of the Association for Computational Linguistics* 4 (2016), pp. 401–415. DOI: 10.1162/tacl_a_00107. URL: https://aclanthology.org/Q16-1029.

[13] J. R. Bormuth. "Cloze test readability: Criterion reference scores". In: *Journal of educational measurement* 5.3 (1968), pp. 189–196.

[14] L. Martin, S. Humeau, P.-E. Mazaré, É. de La Clergerie, A. Bordes, and B. Sagot. "Reference-less Quality Estimation of Text Simplification Systems". In: *Proceedings of the 1st Workshop on Automatic Text Adaptation (ATA)*. Tilburg, the Netherlands: Association for Computational Linguistics, Nov. 2018, pp. 29–38. DOI: 10.18653/v1/W18-7005.

[15] R. Flesch. "Marks of readable style; a study in adult education." In: *Teachers College Contributions to Education* (1943).

[16] R. Gunning et al. "Technique of clear writing". In: (1952).

[17] G. H. Mc Laughlin. "SMOG grading-a new readability formula". In: *Journal of reading* 12.8 (1969), pp. 639–646.

[18] R. Senter and E. A. Smith. *Automated readability index*. Tech. rep. CINCINNATI UNIV OH, 1967.

[19] M. Coleman and T. L. Liau. "A computer readability formula designed for machine scoring." In: *Journal of Applied Psychology* 60.2 (1975), p. 283.

[20] G. R. Klare. "Assessing readability". In: *Reading research quarterly* (1974), pp. 62–102.

[21] E. Dale and J. S. Chall. "A formula for predicting readability: Instructions". In: *Educational research bulletin* (1948), pp. 37–54.

[22] S. V. Balakrishna. "Analyzing Text Complexity and Text Simplification: Connecting Linguistics, Processing and Educational Applications". In: ().

[23] M. Vogel and C. Washburne. "An objective method of determining grade placement of children's reading material". In: *The Elementary School Journal* 28.5 (1928), pp. 373–381.

[24] D. S. McNamara and A. C. Graesser. "Coh-Metrix: An automated tool for theoretical and applied natural language processing". In: *Applied natural language processing: Identification, investigation and resolution*. IGI Global, 2012, pp. 188–205.

[25] E. Fry. "A readability formula that saves time". In: *Journal of reading* 11.7 (1968), pp. 513–578.

[26] K. Rayner. "Eye movements in reading and information processing: 20 years of research." In: *Psychological bulletin* 124.3 (1998), p. 372.

[27] G. G. Hruby and U. Goswami. "Neuroscience and reading: A review for reading education researchers". In: *Reading Research Quarterly* 46.2 (2011), pp. 156–172.

[28] J. Medero and M. Ostendorf. "Atypical prosodic structure as an indicator of reading level and text difficulty". In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2013, pp. 715–720.

[29] S. S. Al-Thanyyan and A. M. Azmi. "Automated text simplification: A survey". In: *ACM Computing Surveys (CSUR)* 54.2 (2021), pp. 1–36.

[30] J. Qiang, Y. Li, Y. Zhu, Y. Yuan, Y. Shi, and X. Wu. "LSBert: Lexical Simplification Based on BERT". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29 (2021), pp. 3064–3076. DOI: 10.1109/TASLP.2021.3111589.

[31] F. Alva-Manchego, L. Martin, A. Bordes, C. Scarton, B. Sagot, and L. Specia. "ASSET: A dataset for tuning and evaluation of sentence simplification models with multiple rewriting transformations". In: *arXiv preprint arXiv:2005.00481* (2020).

[32] L. Richardson. "Beautiful soup documentation". In: *April* (2007).

[33] J. Devlin, M. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018).

[34] I. Chalkidis, M. Fergadiotis, P. Malakasiotis, N. Aletras, and I. Androutsopoulos. "LEGAL-BERT: The muppets straight out of law school". In: *arXiv preprint arXiv:2010.02559* (2020).

[35] D. Kauchak, G. Leroy, and A. Hogue. "Measuring text difficulty using parse-tree frequency". In: *Journal of the Association for Information Science and Technology* 68.9 (2017), pp. 2088–2100. DOI: 10.1002/asi.23855.

[36] J. Kohl. "Improving Translatability and Readability with Syntactic Cues." In: *Technical Communication: Journal of the Society for Technical Communication* 46 (1999).

[37] L. Martin, É. de la Clergerie, B. Sagot, and A. Bordes. "Controllable Sentence Simplification". English. In: *Proceedings of the 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, May 2020, pp. 4689–4698. ISBN: 979-10-95546-34-4. URL: https://aclanthology.org/2020.lrec-1.577.

[38] S. Skylaki, A. Oskooei, O. Bari, N. Herger, and Z. Kriegman. *Named Entity Recognition in the Legal Domain using a Pointer Generator Network*. 2020. arXiv: 2012.09936 [cs.CL].

[39] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. "Exploring the limits of transfer learning with a unified text-to-text transformer". In: *arXiv preprint arXiv:1910.10683* (2019).

[40] A. Bražinskas, M. Lapata, and I. Titov. "Few-shot learning for opinion summarization". In: *arXiv preprint arXiv:2004.14884* (2020).

[41] E. Zolotareva, T. M. Tashu, and T. Horváth. "Abstractive Text Summarization using Transfer Learning." In: *ITAT*. 2020, pp. 75–80.

[42] A. R. Fabbri, S. Han, H. Li, H. Li, M. Ghazvininejad, S. Joty, D. Radev, and Y. Mehdad. "Improving zero and few-shot abstractive summarization with intermediate fine-tuning and data augmentation". In: *arXiv preprint arXiv:2010.12836* (2020).