I Have(n't) Read And Agree To The Terms Of Service

Stanford CS224N Custom Project

Hannah Zhang
Department of Computer Science
Stanford University
hzhang16@stanford.edu

German Enik
Department of Computer Science
Stanford University
german.enik@stanford.edu

Varun Tandon
Department of Computer Science
Stanford University
varunt@stanford.edu

Abstract

People interact with legalese on a daily basis in the form of Terms of Service, Cookie Policies, and other agreements that (for the most part) are mindlessly accepted. In order to give people transparency into the services they use, we aim to develop an abstractive summarizer which simplifies these legal documents. Leveraging an existing dataset of human interpretations of Terms of Service, known as TOS;DR, we developed a two-step pipeline involving extractive summarization followed by text simplification. We use state-of-the-art, transformer-based methods for both steps of this pipeline. Specifically, for extractive summarization we use the CNN/DM BertExt model presented by Liu et al. For text simplification, we used the ACCESS model presented by Martin et al. We first established a baseline of our extractive summarizer performance via ROUGE, our text simplifier performance via SARI and FKGL, and the performance of our pipeline end-to-end via ROUGE. These baselines were established using the pretrained models provided by the authors of these papers. We then employed a variety of data cleaning and data augmentation techniques, as well as model finetuning to improve upon these baseline results. We demonstrate that these techniques result in an improvement in model performance on legalese, both when these models are used in isolation, as well as when these models are used end-to-end. While we did improve on our baseline results, our qualtiative analysis demonstrates that these models are not near the level required for a production-level system, and our results demonstrate shortcomings in the adaptability of these large transformers when pretrained on specific, parallel datasets.

1 Key Information to include

- Mentor: John Hewitt

2 Introduction

Millions of people without proper training have to interact with legalese, a highly professional dialect of English that is used in contracts and other legal settings. This dialect is only partly mutually intelligible with mainstream English dialects. A very common example of legalese in the common public sphere is the Terms and Conditions of Service that almost everybody skips and agrees to by default. Recent conspiracies (like changes of PII usage by Facebook) and initiatives (like Apple requiring all apps be extra transparent about their PII usage via "Privacy Labels"), as well as continuing trends of businesses shifting to the online setting (which is being sped up by the ongoing pandemic) strongly motivate this research paper. Our goal is to combine and modify current neural models that perform text simplification and text summarization to build a legalese-to-laymen English machine translation system for Terms and Conditions of Service agreements. We will be evaluating the performance of transformers on this task. Our task is to translate Terms of Service excerpts from legalese to mainstream English. An input would be an original Terms of Service taken directly from a website, while an output would be a shortened summary of the Terms of Service in mainstream English.

3 Related Work

The first paper we considered was by Anand & Wagh: "Effective deep learning approaches for summarization of legal texts". This paper implements a method of text simplification via RNNs and CNNs for binary classification of sentences in legal documents as important or unimportant [1]. Since this was one of the few summarizers we found that was applied to legal documents, we were intrigued by this approach, and curious if we could apply transformers to a similar task.

Another paper of importance that we considered was by Weng, Chung, & Szolovits, "Unsupervised Clinical Language Translation". This paper was one of the works that inspired us to pursue abstractive summarization. The authors built a model to express medical documents (usually written in highly professional medical jargon), for regular patients that do not speak the medical dialect. The paper used an unsupervised approach by aligning highly dimensional subspaces

of professional and laymen word embeddings [2]. This paper did not employ transformers which we really wanted to try.

After reading these papers and others, we settled on the two-step pipeline approach, with step one being an extractive summarizer and step two being a text simplifier. We knew we wanted to leverage transformers for this, so we began to look for papers on transformer-based models for completing these tasks.

For our extractive summarizer, we referred to the paper by Liu & Lapata, "Text Summarization with Pretrained Encoders". This transformer based approach had demonstrated strong results on CNN and Daily Mail text simplification, and we decided to use this model (called "CNN/DM BertExt") as a baseline for our extractive summarization [3].

For our text simplifier, we referred to the paper by Martin et al, "Controllable Sentence Simplification". This transformer-based text simplifier based on neural text generation techniques showed promising results on the Wikilarge dataset. Furthermore, the paper was specifically designed to expose some model explainability in the form of "explicit control tokens" which allowed for control over length, amount of paraphrasing, lexical complexity and syntactic complexity. We used this paper's pretrained "ACCESS" model to establish our baseline results [4].

4 Approach

We have a two-step process: (1) summarize the full Terms of Service and similar policies (ToS for brevity) to extract key information and (2) simplify the summarized information to be expressed in layman's terms. Our intuition is that using two, more specialized transformer-based models (one for extractive summarization and one for text simplification) will yield positive results. Furthermore, via this approach, we are evaluating the current state of transformer performance on noisy legalese.

4.1 Approach to Extractive Summarization

We used pretrained BERT for summarization as a starting point and finetuned for ToS language using our data. Liu & Lapata stack several inter-sentence transformer layers on top of BERT for capturing document-level features, and terminates with a sigmoid classification of sentence importance. Their model was pretrained on non-legal news article data with each article's summary as references [3]. We finetune the model on a variety of differently formatted scraped full ToS and their corresponding quotes pulled out as most important by ToS;DR editors. For this part of the pipeline, we focused most of our attention on training data formatting (i.e. experimenting with data augmentation) as scraped terms of service were not clean and first few experiments were showing poor results. See Data and Experiments sections for more details.

We calculated the baseline by evaluating the performance of Liu & Lapata's model's extracted summary quotes from full ToS, using those documents' corresponding ToS;DR quotes as reference. Because we used a pretrained checkpoint as a base for all of our fienetuning, we did not edit the architecture of the model – adjusting it would make the pretrained weights pointless because gradient backpropagation would change. Therefore, all the used PyTorch code was written by Liu and Lapata (see references).

Lapata & Liu offered Transformer & BERT models pretrained on various datasets; we chose "CNN/DM BertExt" (trained on CNN and DailyMail articles) by examining their performances in the paper. More concretely, "CNN/DM BertExt" performed second best based on the ROUGE metric in Liu & Lapata's experiments; it was trained on twice as much data as the best performer; and CNN/DM data were more extractive than other datasets used for training (like XSum). Since we used the pretrained model for general understanding of language and finetuned on our own (that differs quite a lot in language used from news articles), we went with "CNN/DM BertExt".

Average article length in Liu & Lapata's training data was $\sim$ 600 words, and "CNN/DM BertExt" only took inputs truncated to 512 tokens. We didn't truncate any ToS documents because they often contain key information beyond 512 characters. Instead, our code split each ToS into chunks, summarized each one, and then joined all summaries to compare against the reference summary. We obtained document d's ith chunk c_i as follows: tokenize d; set c_i to a substring of d starting at c_{i-1}'s end index and of length 512; if a newline character is present in c_i, make c_i's last newline character its last token and return; else, repeat the previous step for a period; else, repeat the previous step for a semicolon; else, return c_i of size 512. We did not count other punctuation characters like "?" or "!" because they are uncharacteristic of our data's genre.

4.2 Approach to Simplification

For text simplification, we opted to use the ACCESS (AudienCe-CEntric Sentence Simplification) model. The ACCESS transformer is trained on the Wikilarge dataset and uses the same architecture in the "Attention Is All You Need" paper [5]. The model used an embedding dimension of 512, fully connected layers of dimension 2048, 8 attention heads, 6

layers in the encoder and 6 layers in the decoder. Moreover, the ACCESS model was designed to allow for explicit control over Sequence-to-Sequence based models, allowing the users of the model to tweak the length, amount of paraphrasing, lexical complexity, and syntactic complexity of the simplifications returned by the model. The author's approach involves using the Fairseq toolkit to train a Transformer model with various hyperparameters [4]. In order to parameterize the Seq2Seq models with the attributes desired (such as length, paraphrasing, etc.), the authors append a control token at the beginning of source sentences.

The explicit control tokens noted in the ACCESS paper are NbChars, LevSim, WordRank, DepTreeDepth [4]. They correspond to the code variables LengthRatioPreprocessor, LevenshteinPreprocessor, WordRankRatioPreprocessor, and DependencyTreeDepthRatioPreprocessor, respectively. To elucidate what the control tokens mean, *NbChars* is the character length ratio between the source sentence and target sentence; word length has been shown to correlate with lexical complexity. The Levenshtein distance between two words is the minimum number of single-character insertions, deletions or substitutions required to change one word into the other. *LevSim*, the Levenshtein ratio, is the Levenshtein distance divided by the length of the longer of the two words. *WordRank* involves comparing frequencies of words in the target sentence with frequencies of words in the source sentence, which has been shown to correlate with word complexity. Lastly, *DepTreeDepth* is the ratio of the maximum depths of the dependency trees of the source and the target, signifying syntactic complexity.

We were able to modify the ACCESS code to take in our own evaluation data, and we were able to evaluate the network trained on Wikilarge on our data. We then modified the ACCESS code to allow for finetuning. Specifically, we load a checkpoint from the pretrained model, and then train the model on our own formatted ToS;DR data. We also used random search to find the optimal parameters. We'll go more in depth in **5.3.2** about the various hyperparameters we changed. Out of the 241 hyperparameter configurations we ended up with, we had 20 that demonstrated a nontrivial improvement on the model performance . We used the most successful hyperparameters to finetune our model for the pipeline. The hyperparameters are also in section **5.3.2**.

4.3 Pipeline

For the pipeline, we first fed a subset of the Terms of Service chunks to the summarization model, generated candidate summaries, and input them into our simplification model with the hyperparameters from our best finetuning results to generate an abstract summary. We started off with a baseline, with no finetuning, then we tried various data input formats for summarization and finetuning methods simplification, which improved our results for both and led to more pipeline results. See the **Experiments** section below for details.

5 Experiments

5.1 Data

All of the code involving parsing or reconfiguring data was written by us. We downloaded and parsed the ToS;DR (Terms of Service; Didn't Read) data. ToS;DR has a .json file for each service, containing quotes taken directly from the service's ToS. Each pair of ToS quotes and their corresponding simplifications forms a "point" [6]. Each point is peer-reviewed and must be approved by other human evaluators. In their .json files, each point consists of:

- verbatim quote: a quote taken directly from a Terms of Service, Privacy Policy, or similar document
- the title given to the point, generally the simplification that the author wrote for the point
- case: the category that the quote is assigned to (e.g. "This service may collect, use, and share location data")
- the name of the source document that the quote was taken from (e.g. "Terms of Use")

We extracted the key data for our purposes: verbatim quotes, human-written and peer-reviewed simplifications, service name (e.g. "Amazon"), and the names of the documents that the quotes are from. We then reformatted the data (e.g. removing HTML data and newlines) and created a CSV file with the information. However, our summarization model needed full text as inputs, so we gathered 770 full texts; 371 were downloaded through links in the ToS;DR dataset, while the rest were manually searched up to grab the links, then the texts were scraped through a script we wrote.

For summarization, out of the 770 full texts and ToS;DR quotes, we were only able to use 115 for most experiments (on which we achieved some of our best results): over time, companies change the terms of service and the pulled out ToS;DR quotes were not contained in a lot of them. We cleaned 15 by hand and 100 algorithmically by filtering out all full terms of service that were missing at least one ToS;DR quote. (As described below, all experiment titles that contain the word *clean* were trained and evaluated on the 115 cleaned data files.) We joined the quotes taken from the CSV so that all quotes from each document of each service were put in the same .txt file. We used the same schema for the full texts we had, to match source documents; quotes and full texts came from the same source document if their filenames were the same. We followed a 60:20:20 split for training, validation, and test sets; upon manual examination of data, we determined pulled-out quotes to all have very similar structures (most contained words like *privacy* or *cookies*), which led us to believe that having slightly less data for training and more for evaluation would results in more accurate evaluation without hurting evaluation.

For simplification, we took out all of the quotes and put them line by line in a .complex file to serve as the input to the model. Similarly, we took out the human simplifications and put them line by line in a .simple file to serve as the evaluation comparison. We used scikit-learn's `train_test_split` to generate a train, test, and validation set with proportions 0.9, 0.05, and 0.05 respectively (5040, 280, and 280 sentences).

5.2 Evaluation method

We are using ROUGE-1, ROUGE-2, and ROUGE-L as primary metrics and human evaluation as secondary to evaluate the performance of extractive text summarization (using precision and recall). ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation and is commonly used to rate automatic summarization and machine translation. -1 and -2 stand for unigram and bigram overlap, and -L stands for LCS (Longest Common Subsequence) overlap between candidate and gold summaries. Liu and Lapata had ROUGE implemented in their repository (using a python package), so we used their scripts to evaluate performance after each validation run.

SARI (System output Against References and against the normal sentence) is a lexical simplicity metric used for evaluating automatic text simplification systems. The metric compares the predicted simplified sentences against the reference and the source sentences [7]. It does so by taking the arithmetic mean of the n-gram F1-scores of addition, deletion, and include operations. SARI rewards words that appear in both the output texts and the references by rewarding operations that aim for that goal; for example, addition operations for simplified words and deletion operations for unnecessary words [7]. The ACCESS authors indicate that for the purpose of sentence simplification, SARI is a better metric than BLEU (since BLEU places too much reward on not changing the sentence structure) [4]. Thus, we will be using the SARI metric to determine the relative performance of our simplification models. SARI is also the current standard for text simplification, given that it weighs specific changes made to the input text and correlates well with human judgements. Furthermore, we will calculate FKGL as a secondary metric to evaluate readability. FKGL stands for "Flesch–Kincaid Grade Level" and is a readability test designed to indicate what US grade level a piece of text corresponds to [8]. In other words, we can use FKGL to look at difficult a passage in English is to understand. Ideally, we want our SARI score to be as high as possible, as it signifies that the words that are added, deleted, and kept are "better", but we want our FKGL to be lower, as a lower FKGL corresponds to text that is easier to read. However, FKGL has to be used in conjunction with SARI, since FKGL doesn't care for meaning preservation.

For evaluating the whole pipeline (which will be an abstractive summarizer), we are using ROUGE scores as well, since Liu & Lapata's work uses ROUGE to also evaluate their 8 abstractive summarizers. We also used human evaluation to look at the semantics of the result.

5.3 Experimental details

5.3.1 Summarization

For text summarization, we didn't edit the architecture of the model in order to be able to use the pretraiend "CNN/DM BertExt" checkpoint as a start. We kept the position embedding dimension of 512 because the model was pretrained on articles of this size (see section **4.1** for details of our input structure). The model had 12 hidden layers of size 768, 12 attention heads, hidden dropout probability of 0.1, and total vocab size of 30522. We wrote a lot of data parsing scripts for all the different experiment setups we describe below, as well as a summarization pipeline script that automated more data preprocessing (3-step process required before each training), training itself, and logging. Validation was kept out of the pipeline.

Our experiments mostly involved data reformatting and fell into the following three categories (see table below for a summary): *full*, *clean*, and *aug*.

The *full0* experiment was conducted on all the data we had – 770 full ToS files (14050 chunk files) and their corresponding ToS;DR quotes. For each company's ToS chunk, we treated all that company's ToS;DR quotes as gold summary references. This approach was flawed because a lot of chunks still had contaminated data and did not mention anything close to reference quotes (read more in **Results** and **Analysis**).

The *clean*, *clean-n*, *clean-nr*, and *clean-r* were run on cleaned data (as described in **5.1 Data**). Suffix -n means that only data with non-empty summaries was used for training, and -r means that the data was cleaned further with a regex by omitting all characters besides `[A-Za-z0-9 \n.,;?]`. We ran -n experiments to see how more similarity to pretraining data impacts results (all CNN/DM data had non-empty summaries), and we ran -r experiments to investigate how much remaining contamination in the dataset was hurting our performance. Note that although we had 2060 clean chunks, experiments postfixed with -n and/or -r were trained on a smaller number of documents because most chunks' references were empty and regex cleaning invalidated some previously valid files.

Finally, experiments prefixed *aug* were trained on 115 cleaned full ToS files (2060 chunks) as well as variable amounts of augmented training data. We pursued data augmentation in hopes of teaching our model to generate empty summaries for some chunks. We generated augmented data chunks by considering all chunk0's of our datasets (beginning of ToS

often contained junk text) and randomly mixing them up. Validation set only consisted of data from the original 115 files.

For your convenience, here is a table summarizing everything written above. Validation sets were randomized each time and varied structurally among some experiments (i.e. full0 vs regex-cleaned data vs the rest), but were always kept exactly identical between a trial's finetuned model and our baseline.

Experiment	num chunks	regex-cleaned	only non-empty summaries	num augmented chunks
full0	14,050	-	-	-
clean	2,060	-	-	-
clean-n	332	-	Y	-
clean-nr	236	Y	Y	-
clean-r	1,420	Y	-	-
aug-1k	2,060	-	-	1,000
aug-10k	2,060	-	-	10,000

5.3.2 Simplification

For text simplification, we left the general ACCESS structure untouched except for where we modified it to fit our dataset. We made substantial changes to the training code and the evaluation code.

When we performed random search to discover which hyperparameters are best, we initially watched our VM and tried to grab our results (which are printed to the console) as soon as each finetuning finished. However, that limited the amount of time we were able to train our models. That's why, for training, we updated the main training file to use our datasets and to allow us to run finetuning continuously without needing to keep an eye on our virtual machines (VMs) while we were training. We also changed the code that calls outside modeling toolkits and packages like fairseq (Facebook AI Research Sequence-to-Sequence Toolkit) and easse (Easier Automatic Sentence Simplification Evaluation), adding parameters to load the model checkpoints we want as well as changing parameters to maximize the success of our finetuning.

The hyperparameters we ended up changing are shown in the table immediately below along with the randomized values we set. ACCESS is unique since it provides us with the opportunity to tweak the explicit control tokens (the preprocessor target ratios in the table). Each control token's value represents the ratio of the control token value of the target sentence over the token value on the source sentence.

Hyperparameter	**Value**
beam	random integer from 2 to 10, inclusive
dropout	random float from 0 to 1
label smoothing	random float from 0 to 1
lr (learning rate)	random float from 1E-5 to 1E-3, logarithmically scaled
LengthRatioPreprocessor target ratio	random float from 0 to 1
LevenshteinPreprocessor target ratio	random float from 0 to 1
WordRankRatioPreprocessor target ratio	random float from 0 to 1
DependencyTreeDepthRatioPreprocessor target ratio	random float from 0 to 1

Once the model has trained, the ACCESS training script provided by the authors automatically attempts to determine the explicit control ratios optimal for the validation set. Thus, these starting ratios were modified by the training script as a part of a final parameter optimization step. We also note that we used starting ratios between 0 and 1 per the recommendation of the authors; however, these ratios can go up to 2 (and indeed some of our experiment parameters exceed a ratio of 1).

5.3.3 Pipeline

For the pipeline, since simplification did not allow more than 512 tokens per input document when generating a simplification, we wrote a script that split each input summary into chunks, generated simplified text from each chunk, then concatenated all of the results into one final simplified document.

The baseline for our pipeline involved running some Terms of Service chunks through the summarization, then the result was put into the simplification, without any finetuning on either model. Then, we took outputs from summarization and input them into simplification (pipeline names correspond to summarization trial names) : *clean-n*, *aug-10k*, *clean*, *aug-1k*, *clean-nr*. We ran those summarization results through the simplifier finetuned on the best hyperparameters to generate more pipeline results.

6 Results

6.1 Summarization

Entries are of format `finetuned score / baseline score`. Asterisk * indicated best results as compared to the baseline.

Experiment	ROUGE-1	improvement	ROUGE-2	improvement	ROUGE-L	improvement
full0	19.44 / 15.88	+3.56	5.00 / 3.60	+1.4	16.95 / 13.87	+3.08
clean	39.04 / 36.74	+2.3	26.72 / 23.08	+3.64	36.00 / 33.30	+2.70
clean-n*	41.61 / 34.92	+6.69	31.20 / 22.27	+8.93	39.28 / 31.26	+8.02
clean-nr	35.69 / 32.02	+3.67	24.35 / 19.67	+4.68	33.15 / 28.69	+4.46
clean-r*	36.05 / 29.50	+6.55	22.93 / 15.01	+7.92	33.15 / 26.69	+6.46
aug-1k*	39.87 / 32.08	+7.79	29.09 / 18.90	+10.19	37.24 / 28.82	+8.42
aug-10k	35.87 / 35.72	+0.15	23.32 / 22.83	+0.49	32.73 / 32.72	+0.01

The table above contains ROUGE scores for some of our best experiments. We were pleased (but also expected) that in each experiment we ran, the finetuned model performed better than baseline, especially in experiments *clean-n, clean-r*, and *aug-1k*.

The success of *clean-n* was a surprise to us because, admittedly, it resulted from a data processing mistake – we never meant to only take data with non-empty summaries into account. We believe it performed well because that trial replicated pretraining data the most – CNN and DailyMail training data never had empty highlights. This shows the importance of wisely selecting a baseline that is as close to one's task at hand as possible.

The good performance of *clean-r*, contrasted with subpar performance of *clean-nr* with respect to its baseline was initially surprising but is now logical. The whole point of regex cleaning was to lower the negative impact of non-English text that contaminated our data. Since *clean-nr* was only trained on chunks with non-empty highlights, a significant majority of them were already as clean as they could get – regex cleaning probably disturbed already good data by stripping potentially important characters we did not think of including into the regex filter. In contrast, *clean-r* was run on contaminated data, which regex helped clean up. Additionally, *clean-nr* had a very low number of training data.

The good performance of *aug-1k* partially proved our hypothesis right – the model needed to see more bad examples it should ignore. The very subpar performance of *aug-10k* is a little odd since the two trials only differ in the amount of augmented data. However, since the point of augmented data was to overpower the model's knowledge gap from pretraining – making decisions on English articles is a lot more similar to making decisions on English ToS text than to on gibberish data – yet, at the same time, preserve the significance of data with non-empty references, the inferior performance of the trial with 10x more junk data can be expected.

We expected *full0* to perform very poorly, as it did. For some chunks, we practically asked our model to predict English text from junk data like links and html, which is implausible.

6.2 Simplification

In Appendix table A2, we listed the 22 trials that successfully finetuned, along with their SARI and FKGL scores.

We first evaluated our model without finetuning, as a baseline. Our baseline SARI and FKGL scores are 26.894 and 10.685, respectively. All of the successful finetuned results can be found in table A2 in our appendix. Our best run was trial 69, with a SARI score of 36.565 and a FKGL score of 8.660. A table with all of its hyperparameters is in Appendix table A3. This SARI score is the maximum we achieved, and we can see that both the SARI and the FKGL score beat the baseline. There didn't appear to be a strong correlation between any of the hyperparameters and the SARI score, but it does appear as though a lower learning rate and higher LengthRatioPreprocessor and LevenshteinPreprocessor values does better than otherwise.

6.3 End to End Pipeline Results

After extensive tuning of our models individually, we finally evaluated these models end-to-end via ROUGE. Specifically, we compare our model baselines end-to-end against our finetuned models. Below is a selection of our results:

Experiment	ROUGE-1	ROUGE-2	ROUGE-L
baseline	11.68	1.03	8.85
pipeline-clean	14.21	1.82	10.82
pipeline-clean-n	18.51	3.19	13.02
pipeline-clean-nr	15.05	2.96	10.68
pipeline-clean-r	32.34	14.18	22.85
pipeline-aug-1k	10.16	1.08	7.74
pipeline-aug-10k	11.66	1.09	8.88

Here we observe that some of our trials made a clear improvement over the baseline results. Particularly, pipeline-clean-r clearly outperformed all other pipelines despite the fact that it was not the top summarization result. Gibberish data (aug-1k and aug-10k) seems to have negative impact on the end-to-end result even though training on aug-1k dataset performed well on purely summarization. However, we also note that qualitatively, much of the results are still incoherent, difficult to understand, or generally incorrect. (This is discussed in further detail in our **Analysis** section). Regardless, these results are still important, as they demonstrate the feasibility of this approach in improving simplification of legalese.

7 Analysis

7.1 Summarization

Upon examining candidates generated by the best trials, most of the extracted English text seems valid and important for the user signing a ToS agreement to take into account. For instance, here are some excerpts *aug-1k* pulled out and the baseline did not (rest of text is substituted with ellipses for the sake of brevity):

- "however , even if we remove the content or information that you posted , we can not completely prevent further use or disclosure of that content or information by others once you have shared it in a publicly available forum..."
- "you are a citizen of one of the countries identified below , you hereby agree that any dispute or claim arising from this agreement shall be governed by the applicable law set forth below , without regard to any conflict of law provisions , and you hereby irrevocably submit to the non - exclusive jurisdiction of the courts located in the state..."
- "this information is anonymous and contains no personally identifiable data .<q>information you elect to submit to the service in a public or quasi - public manner may be disclosed to other users of the application or the general public (" user submission (s) ")"

Although the finetuned and baseline models did agree on the most important sentences to extract, the best-performing finetuned models seemed to pay extra attention to PII and legal procedures, which is consistent with our in-person review of ToS;DR quotes.

One error category that is persistent across all trials we have conducted is the finetuned model's failure to not pull anything out when presented with a text chunk. This was a significant difference between pretraining data and finetuning data: an average ToS was ~ 15 times longer than CNN/DM articles – a lot of chunks simply had nothing important. Some architectural changes may be necessary to encourage 0-length summaries from the model.

7.2 Simplification (and End-To-End Result Analysis)

An examination of the simplification model clearly demonstrates the shortcomings of attempting to adapt the transformer trained on Wiki text to legalese. Here, we highlight some of the key issues we found in our results, with specific examples. These examples come from our simplifier being run on the output of the summarizer. In other words, the "Simple Text" presented here is equivalent to the results of our end-to-end pipeline. Unfortunately, most of these issues were not issues that were resolvable by the explicit control tokens exposed in ACCESS (which can only control length, amount of paraphrasing, lexical complexity and syntactic complexity).

Complex Text (summarization output)	**Simple Text (simplification output)**	**Issues**
qall rights not expressly granted in the terms are reserved to dailymotion and its licensors	The word " q " comes from the Latin word for " q " , which means " all rights " .	hallucinating information, comprehensibility
we may use information we collect to detect , investigate , and prevent activities on our sites that may violate our terms of use , could be fraudulent , violate copyright , or other rules or that may be otherwise illegal , to comply with legal requirements , and to protect our rights and the rights and safety of our users and others	For example , in the United States , there are many other rules that are illegal in the United States , such as the United States , the United States , Australia , the United States , the United States , the United States... ("the United States" repeated for more times)	hallucinating information, repetitive, comprehensibility
<q>we have never , and will never sell or intentionally make public , any personal information gathered on this site	This means that a person will never sell it , so it will never sell it for a long time , or it will never sell .	repetitive, comprehensibility
<q>the site hosts third - party content	It is the home of the third party .	contextual misunderstanding
<q>when you download and use our services , we automatically collect information on the type of device you use and operating system version	If you download and use our services you can use it to make itpossible to collect information on the computer .	awkward structure

7.2.1 Hallucinations

The error of hallucinating information is a common error described in other papers, including the BART summarizer by Lewis, et al. We believe that the cause for these "hallucinations" is the model attempting to convey some common information found in the training dataset. In fact, some phrases presented by the model such as "in the United States", "comes from the Latin word for", and "comes from the Greek word for" were incredibly common in both our simplifications and the Wikilarge dataset. (see **A1** for more information).

7.2.2 Repetition

This is another common error that manifests in language modeling. This is extensively studied in a paper by Holtzman et al, and their research demonstrates that beam search frequently results in the model assigning high probability to repetitive text [9]. We believe that this could be mitigated via some form of frequency penalty in the text generation process.

7.2.3 Contextual Misunderstanding and Incomprehensibility

These are by far the most common errors that manifest in our predictions, and we believe that to a large extent this has to do with information shown to the transformer in pretraining. We believe that finetuning is limited by the data exposed to the transformer during pretraining. The Wikilarge dataset used in pretraining is (for the most part) incredibly clean data, with comprehensible, human-readable data [10]. This is a stark contrast to our data, which has a large amount of non-English HTML and Markdown snippets. Furthermore, the Wikilarge dataset, which is sourced from Wikipedia, is written a very particular style that is distinct from the legalese we examine. We believe that if this transformer were pretrained on legal data (of which there is limited simplified text), the architecture performance would see significant improvement.

The TOS;DR simplifications were a very specific style of simplification: parsing out information that is relevant to human usage of the service. This is a stark contrast from the style of simplifications in Wikilarge, and we believe this contrast in simplification styles also poses a challenge for this model's improvement.

8 Conclusion

In this project we present the performance of state-of-the-art transformer-based models on abstractively summarizing legalese in the context of Terms of Service. In particular, we present the capabilities of an extractive summarizer and text simplifier working in tandem. While we are able to summarize and simplify some text, for the most part, these models are not capable of outputting key information from these Terms of Service with high consistency. The most common errors we examine involve hallucinations, repetition, contextual misunderstanding, and incomprehensibility. We believe that these errors can be mitigated somewhat by expanding the dataset used for pretraining, in particular pretraining on legal data. Unfortunately, no such dataset is currently available, and would require a sizeable human effort to compile, given that the methods presented are dependent on parallel data. We believe that this work has demonstrated some of the limitations of the very common pretraining approach to using transformers, and in particular, in the most common dataset used, Wikilarge. Moreover, our work demonstrates some of the limitations of pretraining on fully comprehensible text, such as Wikilarge, when the real world contains unintelligible text. Through this process, we learned a lot about working with techniques from the class in real-world situations, specifically diving in to managing large files across cloud virtual machines, hyperparameter tuning, and data cleaning and parsing.

References

[1] D. Anand and Rupali Wagh. Effective deep learning approaches for summarization of legal texts. *Journal of King Saud University - Computer and Information Sciences*, 12 2019.

[2] Wei-Hung Weng, Yu-An Chung, and Peter Szolovits. Unsupervised clinical language translation. *CoRR*, abs/1902.01177, 2019.

[3] Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. *CoRR*, abs/1908.08345, 2019.

[4] Louis Martin, Benoît Sagot, Éric de la Clergerie, and Antoine Bordes. Controllable sentence simplification. *CoRR*, abs/1910.02677, 2019.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[6] N/A. Terms of service; didn't read. `https://tosdr.org/`.

[7] Optimizing statistical machine translation for text simplification. volume 4, pages 401–415, 2016.

[8] J. Peter Kincaid, Robert Fishburne Jr., Richard Rogers, and Brad Chissom. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. 1975.

[9] Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *CoRR*, abs/1904.09751, 2019.

[10] Xingxing Zhang and Mirella Lapata. Sentence simplification with deep reinforcement learning. *CoRR*, abs/1703.10931, 2017.

A Appendix

Common Phrase	**Frequency**
""""""""	92
the united states the united states the united	36
" comes from the greek word for "	23
" comes from the latin word for "	18
to make it_easier to make it_easier to make	15
a person or a person or a person	12
q is a term used to describe a	11
do not know what you want to do	11

A1. Most Frequent Phrases in Simplified Text (clean-nr, 925 sentences)

Trial	beam	dropout	label smoothing	lr	LengthRatio	Levenshtein	WordRank	DepTreeDepth	SARI	FKGL
baseline									26.894	10.685
2	8	0.70	0.99	3.72E-05	0.77	0.70	0.15	0.77	35.39	5.06
6	8	0.23	0.65	7.16E-06	0.72	0.81	0.16	0.005	35.72	9.85
13	6	0.67	0.23	0.0004	0.96	0.82	0.71	0.09	36.35	4.93
47	4	0.01	0.08	6.40E-06	0.60	0.98	0.65	0.59	36.21	7.01
48	10	0.97	0.72	0.0002	0.80	0.84	0.01	0.61	35.28	4.47
50	2	0.47	0.55	7.65E-06	0.60	0.97	0.07	0.85	35.13	5.27
60	10	0.25	0.97	0.0005	0.60	0.68	0.27	0.55	35.45	5.98
67	2	0.67	0.82	6.59E-05	0.83	0.80	0.07	0.21	35.86	9.74
68	5	0.07	0.02	4.70E-06	0.53	0.97	0.67	0.63	35.33	4.26
69	2	0.12	0.32	4.35E-06	0.84	0.75	0.65	0.93	36.57	8.66
77	7	0.52	0.68	9.17E-06	0.999	0.69	0.60	0.04	34.64	7.92
82	3	0.36	0.57	2.48E-05	0.98	0.82	0.92	0.30	36.25	9.56
83	9	0.87	0.08	8.12E-06	0.78	0.77	0.05	0.16	36.30	9.12
92	3	0.64	0.13	9.77E-06	0.59	0.90	0.04	0.55	35.58	10.67
105	6	0.80	0.04	4.50E-05	0.94	0.61	0.43	0.37	35.79	8.57
110	6	0.81	0.29	0.0005	0.06	0.24	0.92	0.54	35.98	8.02
112	6	0.87	0.88	6.74E-06	0.90	0.67	0.93	0.83	36.41	5.93
120	2	0.53	0.33	8.02E-06	0.51	0.78	0.67	0.04	35.87	6.60
129	7	0.94	0.74	0.0009	0.70	0.83	0.47	0.96	35.44	12.38
130	10	0.92	0.05	7.91E-05	0.94	0.91	0.91	0.18	35.42	11.73

A2. Hyperparameters and Evaluation Scores for Successful Simplification Finetunes
Generally all values are rounded to the nearest hundredth

Hyperparameter	Value
arch	transformer
warmup updates	4000
parameterization budget	256
beam	2
dataset	simplification
dropout	0.1196058647
fp16	false
label smoothing	0.3182347886
learning rate	4.35E-06
learning rate scheduler	fixed
max epochs	100
max tokens	5000
metrics coefficients	[0, 1, 0]
optimizer	adam
validations before SARI early stopping	10
LengthRatioPreprocessor target ratio	0.8402369984
LevenshteinPreprocessor target ratio	0.7460899603
WordRankRatioPreprocessor target ratio	0.6479291499
DependencyTreeDepthRatioPreprocessor target ratio	0.9258573112
SentencePiecePreprocessor vocab size	10000

A3. Hyperparameters for Simplification Random Search Trial 69