School of Electronic Engineering and Computer Science

MSc Big Data Science
ECS751P
Project Report 2019

Sentence Compression for Improved Understanding of Legal Texts

Richard Batstone

August 2019

Abstract

Law touches on almost all aspects of our lives. However, our ability to comprehend law is low: the structure, style and language of legislation makes it impenetrable to the general reader. Given the different audiences that legislation must serve, it seems unlikely that 'plain English' initiatives alone will produce legislation that a general reader can understand.

Instead, we develop computational methods for the automatic compression of legislative language, with the aim of increasing its readability. Automatic compression of legislative language is a novel task and, as a necessary part of this work, we contribute a parallel corpus of legal rule compressions to form the basis of statistical or rule-based systems.

We explore and compare numerous methods of applying rules-based and statistical compression models to the legislative domain. We describe the three highest performing models, each generating encouragingly convincing compressions:

1. a high-precision rule-based model achieving a compression ratio of 0.90 (i.e. removing, on average, 10% of tokens in the input sequence, with an F1 score of 0.77);

2. a 'BERT' and rules-based ensemble model trained only on the legislative corpus, achieving a compression ratio of 0.72 (closest to the target ratio of 0.63), with an F1 score of 0.57; and

3. a 'BERT' and rules-based ensemble model pre-trained on a corpus of news headline compressions before being 'fine-tuned' on the legislative corpus, achieving a compression ratio of 0.39 and F1 score of 0.42.

We suggest avenues of future work including developing these ensemble models, or training large-scale language models for use in the legal domain.

Declaration

This thesis, with any accompanying documentation and implementation, is submitted as part of the requirements for the degree of *MSc Big Data Science* at Queen Mary University of London.

It is the product of my own labour except where indicated in the text. This thesis may be freely copied and distributed provided the source is explicitly acknowledged.

London, 21 August 2019

Richard Batstone

Acknowledgements

First and foremost, I would like to thank my thesis supervisor Mehrnoosh Sadrzadeh for her calm and thoughtful guidance over the last year. I am particularly grateful for the freedom and support she have given me to explore my interests.

I would also like to thank my mentors and managers at Slaughter and May for their support in allowing me the time to pursue this degree. Few people are lucky enough to study their academic interests whilst still enjoying rewarding professional work, and I count myself very fortunate.

Of course, my enduring thanks go to my partner Charlotte, not least for her patience and understanding in being (indirectly) subjected to the vagaries of the academic calendar.

Contents

Chapter 1

Introduction

1.1 Motivation: understanding legal rules

We are, as a society, prolific rule-makers. Each year some tens of public general acts and thousands of statutory instruments are added to the statute book.[1] These rules impact almost every area of our lives, from employment rights and planning applications to pensions and social care.[2]

Whilst digitisation has allowed many more people to access the law, access alone does not equate to understanding (Curtotti and McCreath [2013]). Successive reports have shown that people's ability to comprehend legislation is low (Office of the Parliamentary Counsel [2013]). And, perhaps surprisingly, it is not just the general public who struggle with legislation: legal professionals and law makers who regularly interact with legal language also struggle to comprehend legislation.

There is no one single cause of this lack of comprehension. In its 2013 report, the Office of the Parliamentary Counsel notes there are a range of factors that contribute to the complexity, and thereby the lack of comprehension, of law, including:

1. the complexity and inter-connectedness of modern life and commerce that naturally leads to complexity in our laws;

[1] Counts of UK public general acts and statutory instruments as made available on Legislation.gov.uk, at http://www.legislation.gov.uk/uksi and http://www.legislation.gov.uk/ukpga, accessed July 2019.

[2] In this thesis, we will primarily discuss 'law' as legislation created by the state. Of course, not all legal rules are created by the state and not all things that fall within the ambit of 'law' can be described as systems of rules: individuals may make rules between themselves in contracts, lawyers may give legal advice during the negotiations and the courts may pass judgments on the interpretation of those contracts when they come into dispute.

2. despite the efforts of multiple governments to carry out 'red tape cutting' initiatives, the volume of new legislation entering the statute book has remained largely consistent in the last 40 years. The exact volume of legislation in force is difficult to estimate, but commentators suggest there are in the order of 50 million words of legislation in force, with 100,000 words added and subtracted every month;[3] and

3. the language, style and structure of legislation is not 'user-friendly' and there is a lack of explanatory materials and guidance.

This thesis is motivated by the third factor. Legislators have aspired to improve the clarity and 'readability' of legal language for many years. The Australian Office of Parliamentary Counsel, for example, first published its 'Plain English Manual' for legislative drafting in 1993.[4] Many of these principles are reflected in the UK's own 'Drafting Guidance'.[5] However, a cursory look at legislation entering the statute book gives an indication of the level of complexity that remains in legislative drafting. For example, consider the following two sections from 'The Designation of Schools Having a Religious Character (England) Order 2019':

> *(1) The following voluntary schools are designated as schools having a religious character— (a) Marston Moreteyne Church of England VC School, Bedfordshire, MK43 0NE; (b) Monksmoor Park Church of England School, Northamptonshire, NN11 2PD; and (c) Sacred Heart Roman Catholic Voluntary Aided Primary School, Norfolk, PE37 7QW.*
>
> *(2) The religious denomination in accordance with whose tenets religious education is, or may be, required to be provided in accordance with Schedule 19 to the School Standards and Framework Act 1998(1), is— (a) at the schools designated in paragraph*

[3] TEDx Houses of Parliament speech by former First Parliamentary Counsel Richard Heaton, https://www.gov.uk/guidance/good-law, accessed July 2019.

[4] https://www.opc.gov.au/publications/plain-english-manual, accessed July 2019.

[5] https://www.gov.uk/government/publications/drafting-bills-for-parliament, accessed July 2019.

(1)(a) and (b), Church of England; (b) at the school designated in paragraph (1)(c), Roman Catholic.

Many readers may have struggled reading section (2), perhaps only understanding (inferring) its meaning when reaching the penultimate line and seeing the references to 'Church of England' and 'Roman Catholic'. Having concluded that section (2) tells us the types of the religious schools mentioned in section (1), many readers will also be left wondering why it was needed, given that 'Church of England' and 'Roman Catholic' are included in the *names* of the relevant schools.

This example hints at the fundamental tension in legislative drafting: legislation has to speak to many different audiences, and those audiences have different needs. Lay readers wish to understand the high-level impact of a rule (but only as it applies to their circumstances), administrators and lawyers demand precision and certainty (regardless of the cost to readability) and legislators wish to see a clear articulation of their policy objectives (with less regard to the interaction with the complex body of rules of which their legislation is a part).

Rather than attempting to simultaneously cater to all of these audiences, we will explore whether automatic computational methods can be used to simplify legislative language, with the aim of providing a 'view' of legalisation more suitable to a lay person's needs.

1.2 Simplification through compression

In this thesis we will explore simplification through compression. The aim of a compression system is, given an input text, to return a simplified version of that text. The simplification is achieved by removing words from the original. For example, given this section from 'Companies Act 2006':

(1) This section applies where— (a) a company makes a distribution consisting of or including, or treated as arising in consequence of, the sale, transfer or other disposition by the company of a non-cash asset, and (b) any part of the amount at which that asset is stated in the relevant accounts represents an unrealised profit.

The system should return a compressed form, perhaps something like:

(1) This section applies where— (a) a company makes a distribution of a non-cash asset; and (b) the asset represents an unrealised profit.

The compressed form is a sub-sequence of the original with ordering retained. The intention is to arrive at a formulation that is more straightforward for humans to comprehend, whilst retaining the key meaning of the original. The relationship between text compression and the broader task of text simplification is discussed in section 2.1 but we note here that compression is an inherently more constrained task than simplification. In the example above, there are only certain deletions that can be made if the resulting text is to remain grammatical. Simplification by compression means we also cannot replace difficult concepts with more straightforward ones. The system cannot output, for example:

(1) This section applies where— (a) a company transfers an asset to its shareholders; and (b) that asset has grown in value, but the company has not yet received the benefit of that value.

Nevertheless, compression can be a powerful tool in achieving simplification and we suggest that in law (where language is often overly verbose) it is a reasonable place to start.

1.3 Structure of this thesis

In section 2 we review the literature on text simplification, methods of sentence compression, the evaluation metrics typically used for these tasks and the tools and techniques used in natural language processing. In section 3 we discuss the linguistic features of legal language and describe the construction of a corpus of parallel compressions of legal rules for use in this thesis. In section 4 we explore three models for compressing legal text:

1. a rules-based system which compresses legal text through a combination of removing particular syntactic 'sub-trees' and phrases matching a set of patterns (section 4.2);

2. a statistical system using 'LSTMs' following Filippova et al. [2015] and applied to the legal domain following the transfer learning techniques described in Howard and Ruder [2018] (section 4.3); and

3. a statistical system based on 'Transformers' and leveraging the scale of the 'BERT' pre-trained language model (Devlin et al. [2018]) to apply it to the legal domain (section 4.4).

As a development to this work, we also explore whether the rules-based and statistical models might be brought together to enhance performance (section 4.4.4).

Finally, we critically discuss the results (sections 4.5 and 5) and suggest areas for future work (section 6.1).

As noted at point 2. above, part of this thesis follows the work of Filippova et al. [2015] in applying an 'LSTM' based system to sentence compression. The principal differences between this work and Filippova et al. [2015] are that:

1. we apply the 'LSTM' system to the legal domain using transfer learning techniques;

2. aditionally, we propose a rule-based and Transformer-based system for sentence compression in the legal domain; and

3. we contribute a parallel corpus of legal rule compressions to form the basis of statistical or rule-based sentence compression systems.

Chapter 2

Literature review

2.1 Automatic text simplification

Text simplification is the general task of rewriting a text in a way that retains its meaning but is more readable by a target audience. There are a range of applications for text simplification, including: (i) reducing complexity as a pre-processing step for other natural language processing (NLP) tasks; (ii) assisting second language learners; (iii) assisting readers with low-literacy or other learning needs (such as dyslexia or autism); and (iv) as a tool to improve communication in domains where clarity is particularly important (such as engineering manuals) (Shardlow [2014]).

The growth of 'Simple Wikipedia' (a version of Wikipedia written using basic vocabulary and shorter sentences, comprising over 140,000 articles) is cited as evidence of the growing demand for simplified texts (Shardlow [2014]). But whilst automatic text simplification has received much academic interest, it remains a very challenging task.[1] One reason automatic text simplification remains challenging is that it combines a number of distinct sub-tasks, each of which might require a significant level of linguistic or world knowledge. For example, Xu et al. [2016] characterises text simplification as a combination of:

1. *splitting*: decomposing long sentences into a sequence of shorter sentences. For example, transforming "*Steve got home and watered the tomatoes*" into "*Steve got home. Steve watered the tomatoes*";

[1] Tutorial presented by Sanja Štajner and Horacio Saggion on 20 August 2018 at COLING 2018, materials made available at http://taln.upf.edu/pages/coling2018simplification/, accessed July 2019.

2. *deletion*: deleting the less important parts of the text, the task of sentence compression as described in the example in section 1.2; and

3. *paraphrasing*: paraphrasing includes reordering, lexical substitutions and syntactic transformations. For example, transforming "*The poem was composed by the renowned artist.*" into "*The famous artist wrote the poem.*". In this transformation, the text is re-ordered so that it is in the active voice ("composed" for "was composed") and less common words are substituted with appropriate synonyms ("wrote" for "composed" and "famous" for "renowned").[2]

Each of these tasks is challenging in isolation. Sensibly combining them to work together in a text simplification system is therefore especially difficult. Xu et al. [2015] identifies two additional challenges facing the text simplification field:

1. there is a lack of parallel data (i.e. pairs of normal and simplified text) on which to build statistical models. Whilst the 'Simple Wikipedia' database seemingly provides a large volume of simplified text, these texts must first be aligned with their normal counterparts and Xu et al. [2015] shows that: (i) these alignments are often incorrect; and (ii) even where they are correct, the simplified text is often not much simpler than the normal text; and

2. the field has yet to agree on a common evaluation metric to assess the quality of simplifications. This point is discussed in more detail in section 2.3.

Given the difficulty of the general task of text simplification, we focus on the discrete sub-task of sentence compression with the aim of making useful progress applying compression techniques to the legal domain and motivating further work to explore general simplification.

2.2 Methods of sentence compression

As discussed above, text compression can be viewed as a discrete sub-task of text simplification. The aim of sentence compression is, given an input sentence, to return a shorter sentence which

[2]Example due to Sanja Štajner and Horacio Saggion, ibid.

retains the key information of the original and which is itself grammatical. Pitler [2010] gives an overview of the work in this area and notes that approaches to sentence compression can be divided into two families.

The first family of approaches is to make edits to the syntactic parse tree of the input sentence, identifying and 'pruning' branches either through the application of hand-written rules, or through supervised learning models which 'learn' which branches to prune on the basis of a training set (e.g. Knight and Marcu [2000]). For example, given the dependency parse of the sentence "*I really like spaghetti with sauce*" (figure 2.1), a compression system might 'prune' the adverbial modifier attached to "like" and the prepositional phrase attached to "spaghetti". That is, it deletes the words "really" and "with sauce", returning the compressed sentence "*I like spaghetti*" (figure 2.2).

Figure 2.1: Dependency parse, original text

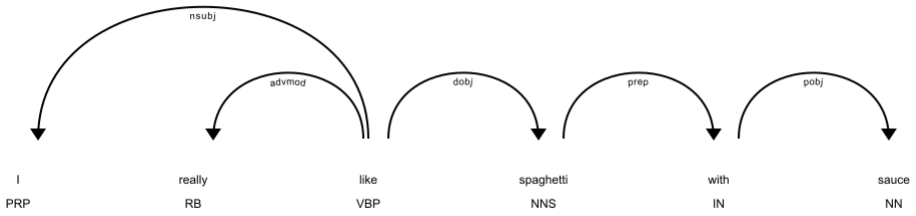
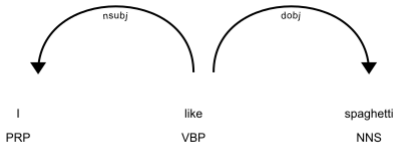


Figure 2.2: Dependency parse, compressed text



One advantage of this family of approaches is that it is easier to ensure the compressed sentences are grammatically correct. However, that is dependent on obtaining an accurate dependency parse, which might be difficult in domains which have particularly complex syntax such as law.

The second family of approaches is to make edits to the sentence string itself. When Pitler [2010] carried out its review, methods in this second family involved extracting features from

words in the input sentence and using these in calculating which of the possible compressions is most probable (e.g. McDonald [2006]). Since that time, the use of 'recurrent neural networks' for sequence-to-sequence modelling tasks have become popular (described further in section 2.4). These models can still be treated as falling within the second family of approaches and the model proposed in Filippova et al. [2015], which we develop in section 4.3, is of this type.

So far, we have described *extractive* sentence compression: the compressed text is a subsequence of the input text. However, for completeness, we note there is also a body of work focused on *abstractive* sentence compression: the compressed text is a proper summary of the input sentence, capable of containing different words, or words in a different order, as compared to the input sentence (see, for example, Chopra et al. [2016]). Work on abstractive sentence compression has tended to be motivated by the task of document summarisation (i.e. the process of automatically producing a summary of one or more extended texts), rather than as a component of text simplification. For that reason, we focus on extractive sentence compression.

2.3 Evaluation metrics

In sentence compression, it is usual (Filippova et al. [2015]) to report the F1 score (i.e. the harmonic average of the precision and recall of the predicted 'labels'), the sentence accuracy (the percentage of compressed sentences that are fully regenerated by the model) and the compression ratio (the number of tokens in the compression divided by the number of tokens in the original sentence).[3] Evaluation of the readability and the informativeness of the compressions is generally carried out manually by human reviewers and we explore a human evaluation task in section 4.6.

However, to aid future work and comparison with more general systems, we will also consider automatic evaluation metrics used in text simplification systems.

[3] Napoles et al. [2011] suggests that compression models should only be directly compared where the models generate similar compression rates, since the more a model deletes, the greater the likelihood that it makes an error.

2.3.1 Flesch-Kinkaid Reading scores

Many text simplification systems use 'readability' metrics to evaluate the quality of their simplifications (Saggion et al. [2015]). Of the many available metrics, the Flesch Reading Ease (FRE) score (Flesch [1948]) remains popular (e.g. Zhu et al. [2010]).

The FRE score combines the number of sentences, words and syllables in a text to assign a readability score. Higher scores indicate texts which are easier to read. The score is given by the formula:

$$FRE = 206.835 - 1.015\left(\frac{\text{total words}}{\text{total sentences}}\right) - 84.6\left(\frac{\text{total syllables}}{\text{total words}}\right) \tag{2.1}$$

Alternatively, the closely related Flesch-Kinkaid Index (FKI) presents a score as a (US) reading level:

$$FKI = 0.39\left(\frac{\text{total words}}{\text{total sentences}}\right) + 11.8\left(\frac{\text{total syllables}}{\text{total words}}\right) - 15.59 \tag{2.2}$$

In the FKI, a lower score indicates higher readability.

Curtotti and McCreath [2013] discusses the limitations of FRE in measuring readability, particularly in the context of legislative language where it has not been validated that FRE is a good indicator of readability. However, given its use in the broader literature we will consider FRE when discussing the general characteristics of legislative language.[4]

2.3.2 BLEU

BLEU (Bilingual Evaluation Understudy score) is a metric originally designed to evaluate the quality of a machine-generated translation by reference to one or more gold standard human translations (Papineni et al. [2002]). However, it has also been widely used in text simplification (Saggion et al. [2015]).

[4]Given FRE is designed to be a corpus level statistic (i.e calculated with reference to total number of words and sentences in a text) we use it for drawing comparisons between legal and non-legal language as a whole and do not report it for the compression model outputs (instead preferring the other metrics discussed below).

BLEU measures the quality of a machine-generated translation (or simplification) by comparing the number of matching n-grams (sequences of 'n' words) in the candidate and reference texts. Matches are not dependent on word order. However, the matching process is modified to ensure that systems which simply repeat frequent words (e.g. 'the') are not over-scored. This is achieved by specifying that a word (or n-gram) in the candidate text is counted as a match only as many times as it appears in the reference text. The final score is the geometric mean of the candidate text's modified precision scores, multiplied by an additional factor which penalises overly short candidates. BLEU typically takes the precision scores for 1, 2, 3 and 4-grams but can be adapted to include larger n-grams. The final score is between 0 and 1. In this thesis, we calculate BLEU score using the implementation in the 'NLTK'[5] library for Python (which calculates BLEU for up to 4-grams) and, in line with standard practice, multiply the score by 100 when reporting.

2.3.3 ROUGE

The ROUGE (Recall-Oriented Understudy for Gisting Evaluation) family of metrics was developed to evaluate the quality of a machine-generated summary of a text by reference to one or more gold-standard human summaries (Lin [2004]). ROUGE is typically used in the sentence compression literature where the compression is abstractive (e.g. Chopra et al. [2016]). Like BLEU, ROUGE measures the quality of a machine-generated summary by comparing the number of matching n-grams in the candidate and reference texts. The key difference between BLEU and ROUGE is that BLEU measures the precision of the n-gram overlaps (i.e. of the n-grams in the candidate, how many were present in the reference) whereas ROUGE measures the recall of the overlaps (i.e. of the n-grams in the reference, how many were present in the candidate). Unlike BLEU, ROUGE does not combine scores for different n-grams and, instead, is reported separately for different n-grams (typically 1 and 2-grams) and for l-grams, where l is the longest common sub-string between candidate and reference texts.[6] In this thesis, we calculate ROUGE using the

[5] https://www.nltk.org/, accessed July 2019.

[6] We note some work also reports a precision and F1 version of ROUGE. In this thesis we report only ROUGE-recall, on the basis that 'precision' quality is captured in BLEU.

'easy-rouge' library for Python.[7]

2.3.4 SARI

Xu et al. [2016] proposed two new evaluation metrics for the task of text simplification: 'SARI' and 'FKBLEU'. SARI has become the principal metric used for evaluating text simplification models.[8] FKBLEU is described in section 2.3.5 below.

SARI (System output Against References and against the Input sentence) is designed to measure the 'goodness' of words that are added, deleted and kept by the simplification system. This is achieved by considering the n-gram precision and recall of additions (n-grams in the candidate which appear in a reference but are not included in the original text), retentions (n-grams appearing in the candidate, original and reference) and the n-gram precision of deletions. The final score is the arithmatic mean of these components.

Note that, unlike BLEU and ROUGE, SARI incorporates information from the original text. In this way, it includes some notion of relative information content between the original text and the compression. Note that the 'addition' component of the score remains relevant for our task, notwithstanding that no new words are added to our sentences. This is because the score measures n-gram additions, and our deletion-based compressions will contain n-grams that were not present in the original text.

In this thesis, we calculate SARI following Xu et al. [2016]'s implementation,[9] modified to use NLTK's tokenisation method to be consistent with our implementations of BLEU and ROUGE.

2.3.5 FKBLEU

The FKBLEU metric, the second metric proposed in Xu et al. [2016], combines a modified BLEU metric with the Flesch-Kinkaid Index (equation 2.2). The modified BLEU metric is the 'iBLEU'

[7] https://pypi.org/project/easy-rouge/, accessed July 2019.

[8] See, for example, commentary on evaluation metrics at http://nlpprogress.com/english/simplification.html, accessed July 2019.

[9] https://github.com/cocoxu/simplification, accessed July 2019.

metric from Sun and Zhou [2012], which is intended to capture the diversity of automatically-generated paraphrases (as well as their adequacy). This score is then multiplied by the sigmoid of the difference in FKI scores of the original and compressed texts.

We note this metric for completeness only and do not report it in this thesis as SARI has become the preferred simplification metric.

2.4 Neural sequence-to-sequence models

2.4.1 Feedforward neural networks

A feedforward neural network (FFNN) is a type of machine learning model for supervised learning. In its broadest terms, an FFNN is a parametric model (with parameters θ) which learns θ^* by solving the optimisation problem:

$$\theta^* = \arg\max_{\theta} \sum_{X,Y} \log p(Y|X;\theta) \tag{2.3}$$

Where the summation is over all pairs of 'training examples' (X,Y), X being the input variable and Y the response variable.

FFNNs are loosely inspired by the human brain and model p through a series of interconnected 'neurons', each of which carries out a simple calculation but which in combination are capable of representing complex functions.

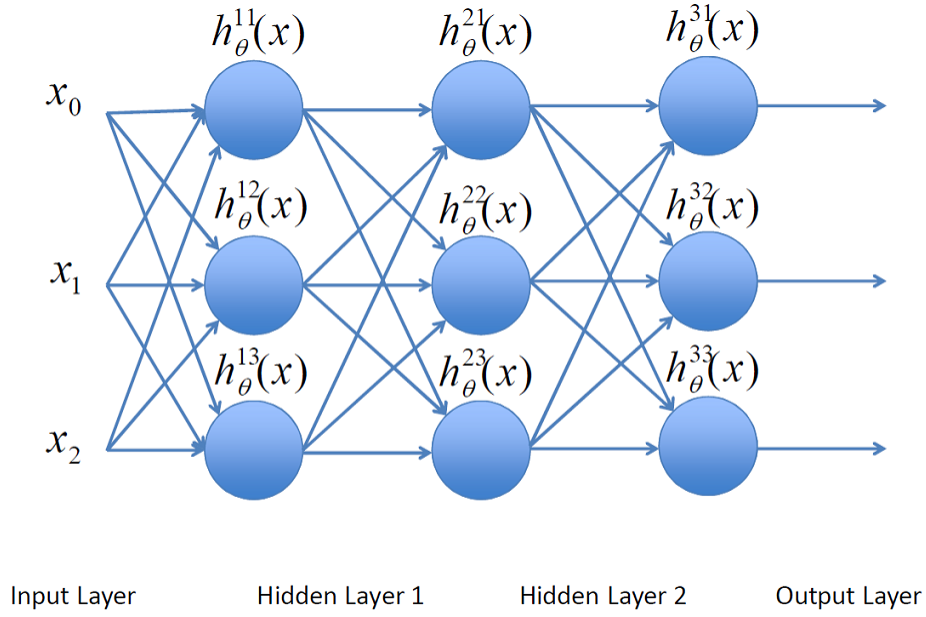
At each neuron, the inputs are linearly combined before being passed to a non-linear 'activation function'. The output $h(\mathbf{x})$ of a neuron is therefore given by:

$$h(\mathbf{x}) = z(\sum_{i=1}^{n} w_i x_i + w_0) \tag{2.4}$$

Where z is the activation function, w_i are real valued 'weights' to be learnt, w_0 is a real valued 'bias' term to be learnt and x_i are the values of the input vector $\mathbf{x}$. The choice of activation function varies. Common choices are the $\tanh x$ and 'rectified linear unit' functions. An FFNN is formed by combining multiple layers of neurons, the input to each layer being the output of the previous layer

(figure 2.3).[10] The final layer of the FFNN is, generally, either the sigmoid function (where Y is a categorical variable) or another linear combination (where Y is a continuous variable). The depth and width of an FFNN (the number of layers and number of neurons in each layer, respectively) are hyper-parameters to be chosen.

Figure 2.3: A feedforward neural network



The parameters θ of an FFNN, the 'weight' and 'bias' vector $\mathbf{w}$ associated with each neuron, are learnt through the gradient-descent by back-propagation algorithm (Rumelhart et al. [1986]). In brief: (i) the parameters of the FFNN are randomly initialised; (ii) predictions for each input variable X are obtained by 'feeding' each variable through the network; (iii) error terms are calculated for each prediction, being some differentiable function of the predicted output and response variable (e.g. the mean-squared error between the prediction and Y); (iv) the partial derivative of each weight term with respect to the (sum of) the error terms is calculated; (v) each weight term is updated by subtracting from it some small fraction of that partial derivative; and (vi) steps (ii) to (v) are repeated until convergence (i.e. the error term stops decreasing).

The 'small fraction' referred to in step (v) is known as the 'learning rate' and is a hyper-

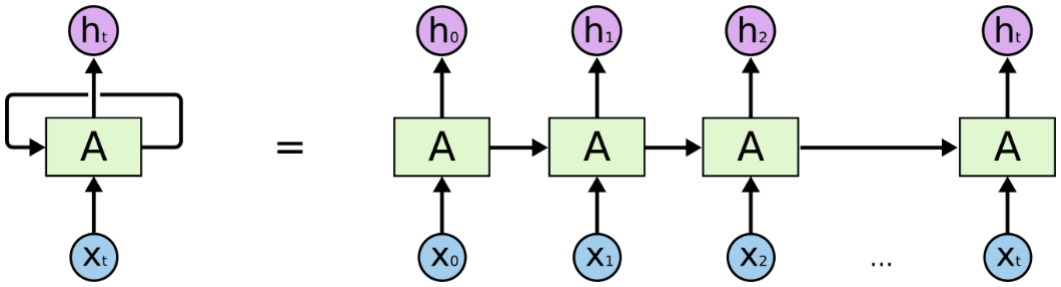
[10] Source: Tim Hospendales, lecture slides - 'Machine Learning', Queen Mary University of London 2017.

parameter of the network to be chosen. In practice, rather than calculating and summing the errors for all training examples before performing a weight update, errors are calculated for small sub-set 'batches' of training examples and the network weights updated after each batch.

2.4.2 Recurrent neural networks

A limitation of the FFNN described in section 2.4.1 is that the input and response variables must be fixed in size. The FFNN is therefore inappropriate for sequence-to-sequence tasks, like sentence compression, where the length of the input sequence varies widely across training examples. A recurrent neural network (RNN) is a generalisation of an FFNN to sequences. In an RNN, a neural network is recurrently applied to the each value in an input sequence. At each step t in the input sequence, the input to the RNN is: (i) the value of the input sequence at that step; and (ii) the 'hidden state' of the RNN calculated at step $t-1$ (i.e. the previous output of the RNN's penultimate layer) (figure 2.4).[11]

Figure 2.4: A recurrent neural network



In this way an RNN can build up a representation of an input sequence: the output of the network at step t contains information from all previous steps. Note that the only difference in the parameterization of an RNN as compared to an FFNN is that the RNN includes a set of learnt 'context' weights: weights that, at step t, are applied to the hidden state received by the network from step $t-1$.

[11] Source: https://colah.github.io/posts/2015-08-Understanding-LSTMs/, accessed July 2019.

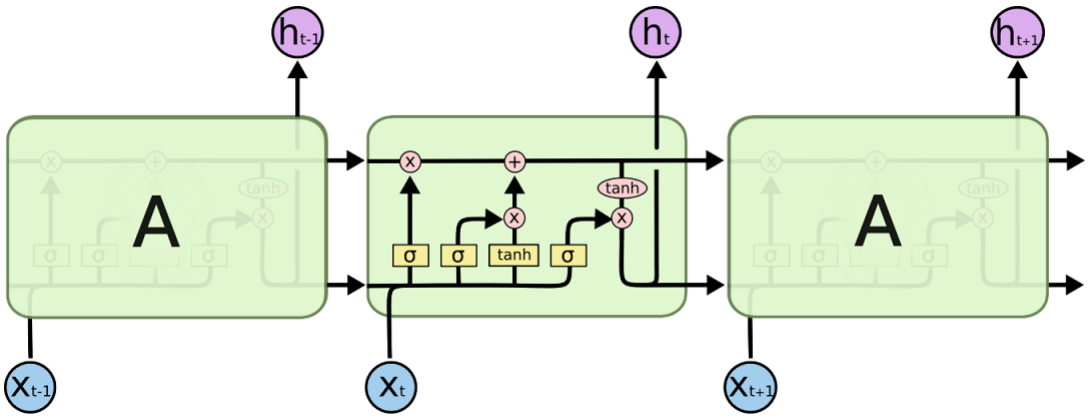
A common application of RNNs in NLP is in language modelling. The task of a language model is to predict the likelihood of observing a given sequence of words in a language. Some sequences of words are quite likely, such as *"The next Thameslink train is cancelled"*, whilst other sequences are very unlikely, such as *"Colourless green ideas sleep furiously"*. RNNs can be trained to model this probability distribution by setting them the task of predicting the next word in a sequence. For example, given as input *"The train arrived at the"* an RNN can output a distribution over a vocabulary. During training, we maximise the probability it assigns to the target word *"platform"*. These methods, together with readily available access to vast quantities of online text, allow powerful language models to be constructed and aspects of these models form the basis of most NLP applications.

Whilst in principle, an RNN can represent information from an arbitrary length sequence, in practice, the output of an RNN at step t is largely dominated by the inputs of the last few preceding steps. This effect arises because, during training, the signal from the target output is quickly diminished as it is applied to previous layers and previous time steps (since applying the chain rule to calculate the partial derivatives for those updates involves multiplying many small numbers together). This effect is known as the 'vanishing gradient' problem and is a key motivator in the design of more sophisticated networks for modelling sequences. A consequence of the 'vanishing gradient' problem is that 'vanilla' RNNs struggle to model dependencies across longer sequences (Bengio et al. [1994]). For example, a 'vanilla' RNN would be unlikely to be able to predict *"platform"* as the next word in the sequence *"I arrived at London Bridge and, despite the horrendous crowds (whose numbers were bolstered by the Saturday football), just made it to my..."*.

One approach to mitigate this challenge is to augment the simplistic RNN with more sophisticated mechanisms to retain information between time steps. The Long Short-Term Memory network (LSTM) is an example of this approach (Hochreiter and Schmidhuber [1997]). The core difference between an LSTM and an RNN is that in addition to the hidden state that is passed between RNN time steps, the LSTM maintains a separate 'cell state' which is designed to retain information across larger numbers of steps. The LSTM achieves this through the interaction of

four layers: (i) a layer to determine what information from the previous cell state should be retained; (ii) two layers to determine the contribution of the current input to the cell state; and (iii) a layer to determine the contribution of the hidden state from the previous time step to the unit's output and new hidden state (figure 2.5).[12] LSTMs are the core of the sentence compression model developed in section 4.3.

Figure 2.5: A Long Short-Term Memory unit

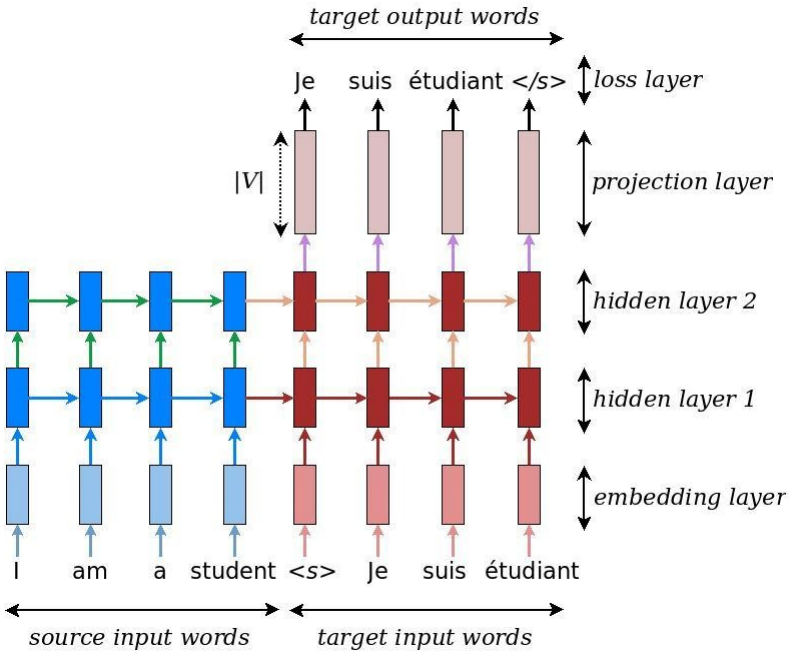


2.4.3 Sequence-to-sequence models

So far, we have discussed representing a sequential input using RNNs. In principle, an RNN can output a prediction at each time step and act as a sequence-to-sequence model. However, in practice, sequence-to-sequence tasks are usually modelled with two RNNs: one to *encode* the input sequence into a fixed length representation, and another to *decode* that representation into a sequential output. This approach has been widely adopted for the task of machine translation (figure 2.6)[13] and is the architecture followed by Filippova et al. [2015], which is developed in section 4.3.

[12] *Ibid.*
[13] Source: `https://github.com/tensorflow/nmt`, accessed July 2019

Figure 2.6: Encoder (blue) and decoder (red) English-to-French translation model



Note that in this architecture, the only information that is shared between the encoder and decoder is the encoder's fixed length representation of the input (usually the hidden state from the encoder's final time step). This leads to an information bottleneck between the two networks: the decoder cannot make direct use of the intermediate hidden states of the encoder. In practice, this limits the lengths of the sequences that can be successfully modelled and refinements to this architecture have been proposed to address this problem.

2.4.4 Transformer neural networks

In order to address the information bottleneck problem in encoder–decoder architectures, Bengio and LeCun [2015b] proposed an 'attention' mechanism to allow the networks to establish direct connections between the input and the target output. The intuition is that, when performing translations, the decoder should be able to focus on (*attend* to) different words in the input at different times. For example, in figure 2.7[14], in decoding the first word in the output, the decoder is able to directly access, and give greater weight to, the encoder's hidden state at the first word of the input.

[14] *ibid.*

In its most basic form, an attention mechanism is calculated (at each step of the decoder) as follows:

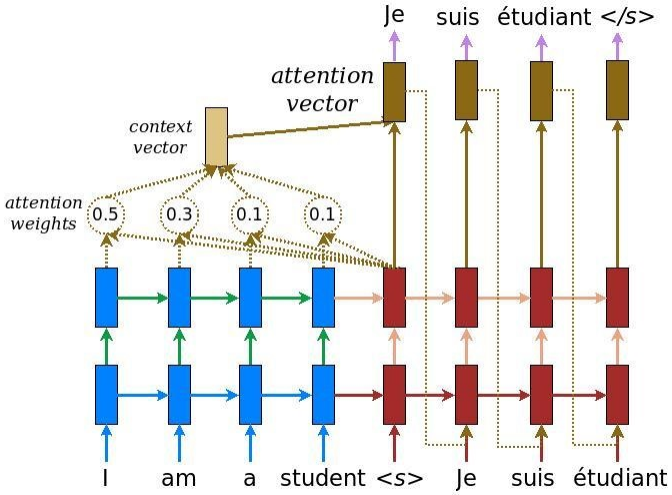
1. the hidden state of the decoder is compared to each of the encoder's hidden states (e.g. dot products between the states are taken), these are the attention scores for the time step;

2. the softmax of the attention scores is taken to give an attention distribution vector for the time step;

3. the attention distribution vector is used to calculate a weighted sum of the encoder's hidden states, giving the attention output; and

4. the attention output is concatenated with the decoder's hidden state and passed to the final layer of the decoder for projection into the vocabulary space.

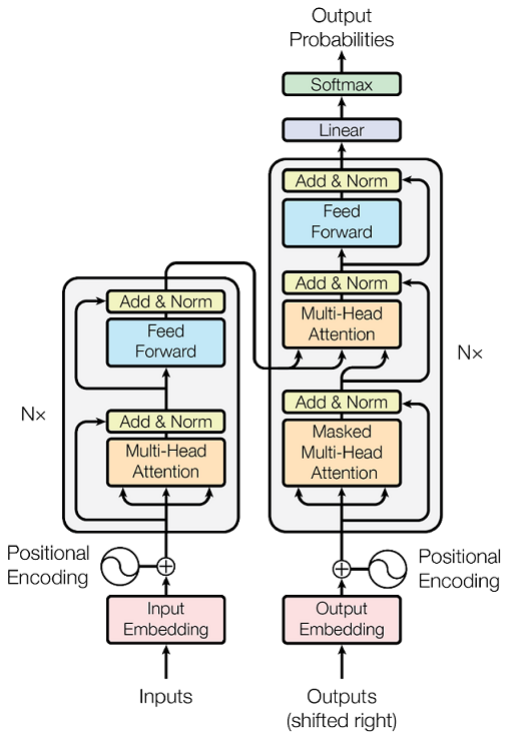
Figure 2.7: Example of an attention mechanism in the first step of a decoder computation



Extending this concept, the 'transformer neural network' (Vaswani et al. [2017]) entirely does away with the recurrent nature of traditional encoder–decoders and instead relies on various forms of attention mechanism (figure 2.8)[15]

[15] Source: Vaswani et al. [2017].

Figure 2.8: Transformer model architecture



In the transformer neural network:

1. the encoder is composed of a stack of identical layers. Each layer includes a multi-head self-attention mechanism and a standard FFNN;

2. in a multi-head self-attention mechanism:

   (a) 'query', 'key' and 'value' vectors are derived for each input token (calculated by multiplying the token embeddings by three learnt weight matrices);

   (b) an attention score for each word is computed by taking the token's 'key' vector and calculating its dot product with the 'querey' vectors for each of the other tokens in the input sequence. The softmax of the attention scores is then multiplied by the 'value' vectors of the other tokens;

   (c) the sum of the attention weighted values gives the self-attention representation for the input token;

(d) linear projection of the 'query', 'key' and 'value' vectors into different dimensions allows the model to jointly attend to information from different representation subspaces (this is *multi-head* attention);

3. each encoder layer is further augmented through a normalisation function and 'residual connection' such that the output of each encoder layer is $\text{LayerNorm}(x + \text{SubLayer}(x))$;

4. the decoder stack is identical to the encoder stack save that each layer also includes a mechanism to calculate attention over the encoder outputs, and the self-attention mechanism is modified to prevent the decoder from attending to 'future' positions in the sequence; and

5. the output layer of the transformer is the softmax of a linear layer.

Since the model does not have a recurrent structure (i.e. each token in the input sequence has its own path through the model), separate positional information is added to the inputs to allow the model to take account of token order. This is achieved by adding a sinusoid to each dimension of the input's embedding:

$$\text{PositionalEncoding}_{(pos,2i)} = \sin(pos/10000^{2i/d}) \tag{2.5}$$

and

$$\text{PositionalEncoding}_{(pos,2i+1)} = \cos(pos/10000^{2i+1/d}) \tag{2.6}$$

Where pos is the position of the token in the sequence, i is the embedding dimension and d is the size of the embedding.

Transformer networks form the basis of the 'BERT' language model, one of current state-of-the-art models described in section 2.5 below.

2.5 Language representations and transfer learning

2.5.1 Language representation

So far, in our description of various statistical models we have referred to the inputs to those models as being sequences of words. Of course, the basic inputs to a language processing model *are* sequences of words, but statistical models do not operate on strings of text and the numerical representation of words (and words in sequence) is a central concern in language processing.

One way of representing words is to treat each word as a 'one-hot' vector indicating the word's position in the language vocabulary. So if the first word in the vocabulary is 'aardvark, 'aardvark' would be represented as a vector with zeroes in every position except the first, where it is one. The length of the vector is equal to the size of the language vocabulary, typically some tens of thousands. The first layer in the network then acts as a map from these sparse vectors to a lower dimensional vector (typically length 150-500) and subsequent layers of the network build up a representation of the word relative to the sequence and the given task.

An immediate consequence of this approach is that the resulting model has a very large number of parameters: at least *vocabulary size* $\times$ *size of embedding layer*. Such models require a very large amount of data to adequately fit these parameters (i.e. to not *overfit*). However, for many supervised learning tasks, there is relatively little training data (perhaps some thousands of examples) and obtaining more data is costly and/or time consuming.

2.5.2 Implicit transfer learning

In order to overcome the training data requirements noted above, it is common to make use of *transfer learning*. In transfer learning we attempt to leverage knowledge about some related task or domain where much more data is available, and apply that knowledge to the target task or domain (Ruder [2019]).

The most common form of transfer learning in language processing is to replace the word vectors learnt in the first layer of the model described above with pre-trained 'word embeddings'.

These word embeddings are obtained by carrying out a variation of a language modelling task (an unsupervised task) on a large corpus of text (typically drawn from online sources): the resulting embeddings are intended to reflect some of the general semantic properties of words from a sufficiently broad domain such that they are useful in other language processing tasks.

Popular methods for calculating word embeddings are the 'skip-gram' and 'continuous-bag-of-words' models (Bengio and LeCun [2013]), both of which seek to leverage the distributional hypothesis of word meaning: words appearing in similar contexts have similar meanings (Firth [1957]). In the skip-gram model, a neural network without a hidden layer is used to predict words appearing in the window around the input word. For example, given the text *"East of the island's capital, Cagliari, beaches suffer from proximity to the city and the SP71 coast road"*, the skip-gram model might be provided with the word 'beaches' and trained to predict the rest of the words in the text. The continuous-bag-of-words model is the inverse of the skip-gram model: it would be provided with the text with 'beaches' removed and trained to predict 'beaches'. The weights of the model become the word embeddings.

In a natural development to using word embeddings, large-scale language models have become popular for obtaining 'deeper' representations of input text. For example, 'BERT' (Bidirectional Encoder Representations from Transformers, Devlin et al. [2018]) is a language model made up of twelve transformer layers and trained on the 'BooksCorpus' and English Wikipedia. In its base version, it has approximately 110 million parameters. Representations of an input text can be obtained from BERT before being passed to a task-specific layer (or the whole model 'fine-tuned' as described in section 4.4).

2.5.3 Explicit transfer learning

Beyond seeking out general purpose representations of words and sentences for use in language tasks, more task-specific transfer learning can be attempted. For example, where we have a model that performs well on a task in a particular domain (e.g. translation from English to French), we would like to leverage that model in improving our performance on the task in a different

domain (e.g. translation from English to Vietnamese). This becomes a necessity when there is comparatively less data available for the task in the target domain.

This *adaptation* of a model for use in a different domain has seen great success in the field of computer vision (Howard and Ruder [2018]). For example, applied computer vision models for identifying particular objects are seldom built from scratch and are, instead, built on the basis of models that have been trained on general object identification tasks. In order to adapt the model to the new domain, it is 'fine-tuned' on the available training data. That is, a small amount of training is carried out on data from the target domain, the aim being to learn something of the distribution of the target domain without overfitting and forgetting the information from the source domain.

Model adaptation has typically been less successful in language processing tasks. However, Howard and Ruder [2018] investigates and proposes general purpose techniques for 'fine-tuning' language models. In brief, the techniques are:

1. *discriminative fine-tuning*: different layers in a model capture different types of information, so should be tuned to different extents;

2. *slanted triangular learning rates*: the learning rate used during fine-tuning should increase to a maximum in order to encourage quick convergence to a region of the parameter space before being reduced to encourage a gradual refinement of parameters; and

3. *gradual unfreezing*: the final layers of the model, which correspondingly capture the least general information, should be fine-tuned before the lower levels of the model.

The application of these techniques is discussed in section 4.3.

Chapter 3

Legal corpora

3.1 Linguistic features of legal rules

In order to establish a baseline level of readability of legislative rules, and to ground comparison with non-legal text, we collect a corpus of legislative text and describe its linguistic features. We note that some existing general language corpora include legal language. For example, the British National Corpus[1] includes legal consultations as well as parliamentary and court proceedings. However, this legal language (whilst interesting) consists of legal advice and oral proceedings, rather than legal rules. Previous work collecting and analysing corpora of legal rules includes:

1. Curtotti and McCreath [2011], which compiled 250 Australian law contracts (approximately 1 million tokens) and compared various linguistic features against a number of news corpora (Brown and Reuters). Among other things, the authors found that the vocabulary of the contract corpus was less diverse than the vocabulary of news articles and general English; and

2. Venturi [2008], which compiled a corpus of eighteen English European Union Directives in consumer law for the purpose of comparing linguistic features of English language legislation with linguistic features of Italian language legislation. Among other things, the authors found that both English and Italian language legislation contain: (i) a higher frequency of prepositional phrases; and (ii) a lower frequency of finite verbal phrases, as compared to

[1] http://www.natcorp.ox.ac.uk/, accessed July 2019.

general language.

3.1.1 Building a corpus of legislative rules

We select 21 UK public general acts as the source of the legislative rules. The acts are selected on the basis of: (i) their 'popularity' among users of Legislation.gov.uk;[2] (ii) the frequency of their use in the author's practice as a solicitor; and (iii) the degree of scrutiny the act received during its passage through parliament.[3] In each case, the intention is to select legislation that has a proven audience.

Legislation is made freely available online[4] and, for ease of processing, can be downloaded in XML format. The XML files are highly structured and the raw text is extracted using the 'Beautiful Soup' library for Python.[5] The unique structure of legislative text immediately raises processing questions. Consider, for example, an extract from the 'Metal Capacity Act 2005' (figure 3.1):

[2] http://www.nationalarchives.gov.uk/about/freedom-of-information/information-requests/list-of-the-10-most-popular-acts-of-parliament/

[3] As indicated by whether the bill's Commons committee stage was taken by the whole House (but excluding consolidation measures and the annual Finance Bill), https://researchbriefings.parliament.uk/ResearchBriefing/Summary/SN05435

[4] http://www.legislation.gov.uk/, accessed July 2019.

[5] https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Figure 3.1: Extract from section 13, 'Mental Capacity Act 2005'

13 Revocation of lasting powers of attorney etc.

(1) This section applies if—

(a) P has executed an instrument with a view to creating a lasting power of attorney, or

(b) a lasting power of attorney is registered as having been conferred by P,

and in this section references to revoking the power include revoking the instrument.

(2) P may, at any time when he has capacity to do so, revoke the power.

(3) P's bankruptcy[F3, or the making of a debt relief order (under Part 7A of the Insolvency Act 1986) in respect of P,] revokes the power so far as it relates to P's property and affairs.

(4) But where P is bankrupt merely because an interim bankruptcy restrictions order has effect in respect of him[F4or where P is subject to an interim debt relief restrictions order (under Schedule 4ZB of the Insolvency Act 1986)], the power is suspended, so far as it relates to P's property and affairs, for so long as the order has effect.

(5) The occurrence in relation to a donee of an event mentioned in subsection (6)—

(a) terminates his appointment, and

(b) except in the cases given in subsection (7), revokes the power.

(6) The events are—

(a) the disclaimer of the appointment by the donee in accordance with such requirements as may be prescribed for the purposes of this section in regulations made by the Lord Chancellor,

(b) subject to subsections (8) and (9), the death or bankruptcy of the donee[F5or the making of a debt relief order (under Part 7A of the Insolvency Act 1986) in respect of the donee] or, if the donee is a trust corporation, its winding-up or dissolution,

(c) subject to subsection (11), the dissolution or annulment of a marriage or civil partnership between the donor and the donee,

(d) the lack of capacity of the donee.

From a structural perspective, note that:

1. syntactically complete sentences might be contained in a single sub-section (e.g. section 13(2)), or be split across a list of provisions (e.g. section 13(1));

2. where a syntactically complete sentence is split across a list of provisions, the introductory wording might contain substantive information (e.g. section 13(5)), or only minimal information (e.g. section 13(6));

3. fragments of sentences contained in lists might stand alone as whole sentences (e.g. section 13(1)(a)) or not make any sense in isolation (e.g. section 13(6)(d)); and

4. information about whether lists of provisions describe cumulative or disjunctive conditions is sometimes contained at the end of the penultimate list member (e.g. section 13(1)(a) and section 13(5)(b)) and is sometimes left to be inferred from the context (e.g. section 13(6)).

These structural features inevitably impact the way in which compression models (and other language processing tools) are applied to legislative text. For example, we must decide whether to analyse and process text as syntactically complete sentences, or as fragments of sentences (i.e. at the lowest sub-section level). We take different approaches for different purposes. In calculating readability and linguistic features for the corpus, we consider legislative text at the level of syntactically complete sentences. In building compression models, we consider legislative text at fragment level.

3.1.2 Analysing the legislative corpus

We use the NLTK and 'spaCy'[6] libraries for Python to analyse the legislative corpus. The corpus comprises approximately 1.25 million words (comparable in size to the Australian Contract corpus collected in Curtotti and McCreath [2011] and the Reuters-21578 corpus[7]). Summary linguistic features for these corpora, as well as the 'news headline' corpus of parallel sentence compressions (introduced in section 3.2), are shown in table 3.1 below:

Corpus	FRE score	Prepositions per thousand words	Type-to-token ratio
Reuters-21578	93.37	91	0.012
Legislation	40.47	157	0.006
News headline	87.69	118	0.020

Table 3.1: Linguistic characteristics of different language corpora

Our findings support those noted by Curtotti and McCreath [2011] and Venturi [2008]: there are significant differences in the linguistic characteristics of legislative and news article language. In particular, legislative language has a higher frequency of prepositional phrases and is less diverse (has a lower type-to-token ratio[8]) than news article language. We also see that legislative language

[6] https://spacy.io/, accessed July 2019.

[7] https://archive.ics.uci.edu/ml/datasets/reuters-21578+text+categorization+collection, accessed July 2019.

[8] Type-to-token ratio is the size of the vocabulary divided by the size of the corpus. We calculate this ratio on the basis of the first 1 million words in each corpus to enable a meaningful comparison.

has a much lower FRE score than news article language, indicating that it is less 'readable'. This is true both of the news article language in the Reuters-21578 corpus and the language of the news headline corpus, which forms the basis of our pre-training task.

3.2 Construction of a parallel corpus of legal rule compressions

As described in section 4, constructing a sentence compression model relies, at least to some extent, on a parallel corpus of pairs of uncompressed and compressed text. The largest such corpus is the 'news headline' corpus constructed by Filippova et al. [2015], which has approximately 2 million examples. That corpus is not publically available but the researchers have made a sub-set of approximately 200,000 examples available.[9] However, we do not expect a model trained only on this news headline corpus to perform well in the legal domain (for the reasons discussed in section 2.5 and particularly in light of the linguistic differences between legislative rules and news article language described above).

Instead, we require a parallel compression corpus for the legal domain. We are not aware of any such corpus and therefore construct one for the purpose of this thesis.[10] The corpus consists of 1,000 compressed and uncompressed pairs of legislative provisions, drawn from the legislation identified in section 3.1.1. The compressions are manually produced based on the author's practice as a solicitor.

The structure of the corpus is summarised in table 3.2 below. The corpus consists of a mix of syntactically complete sentences and sentence 'fragments'. Where fragments of a sentence are included, the syntactically complete sentence (being the concatenation of the sentence fragments making up that provision) is also included. An example of this is shown in table 3.3 below. There is necessarily some repetition between these syntactically complete sentences and the sentence fragments and the corpus is labelled so that either the complete sentences or the fragments can be

[9] https://github.com/google-research-datasets/sentence-compression

[10] The constructed corpus and related scripts are publically available at https://github.com/richardbatstone/legal_language_compression/tree/master/data/raw_data

isolated.

Field name	Type	Description
Reference	String	The legislative reference for the text.
Original text	String	The uncompressed legislative text. Section numbers are removed except where the text is a concatenation of sub-sections, in which the sub-section numbers are retained.
Compressed text	String	The compressed version of the legislative text.
Targeted	Boolean	Whether the example belongs to the targeted training / validation sub-set.
Fragment	Boolean	Whether the example is a fragment or a syntactically complete sentence.
Concatenation	Boolean	Whether the example is a concatenation of a list of provisions.

Table 3.2: Legislative compression corpus data structure

There are 526 sentence fragments and 474 syntactically complete sentences in the corpus. 173 of the syntactically complete sentences are concatenations of provisions. 742 of the examples are marked as 'targeted' examples for training and validation purposes (as discussed in section 4.4.4).

Original text	Compressed text	Targeted	Fragment	Concat.
Case 4 is where— (a) R requests, agrees to receive or accepts a financial or other advantage, and (b) the request, agreement or acceptance itself constitutes the improper performance by R of a relevant function or activity. (*Section 2(3) Bribery Act 2010*)	Case 4 is where— (a) R accepts a financial or other advantage, and (b) the acceptance itself constitutes the improper performance by R of a relevant function.	No	No	Yes
R requests, agrees to receive or accepts a financial or other advantage, and (*Section 2(3)(a) BA 2010*)	R accepts a financial or other advantage, and	Yes	Yes	No
the request, agreement or acceptance itself constitutes the improper performance by R of a relevant function or activity. (*Section 2(3)(b) BA 2010*)	the acceptance itself constitutes the improper performance by R of a relevant function.	Yes	Yes	No

Table 3.3: Example fragment and complete sentence compressions

The corpus is not balanced in the sense that it contains only minimal examples of provisions which are not capable of compression.[11] This can be seen in the FRE score for the corpus which, for the uncompressed examples, is only 10.29 (whereas for legislation as a whole we would expect a figure of around 40, see section 3.1.2). That is, in only selecting provisions which are candidates for compression, those provisions are naturally more complex (have a higher FRE score) than legislative text as a whole. The FRE score for the compressed examples is 39.23.

3.3 Modes of compression

The compressions were carried out intuitively, without any pre-conceived rules in mind. Nevertheless, some common themes emerge and we discuss these briefly as they provide an insight into the construction and evaluation of the compression models.

The broad themes are:

1. removal of synonymous or hyponymous words and concepts;
2. removal of reference words and phrases where the reference can be assumed from the context; and
3. removal of other extraneous information.

3.3.1 Synonymy and hyponymy

A common objective of legal rules is to be as specific and unambiguous as possible. This objective may lead to the elucidation of a number of connected concepts or words where, in non-legal language, a single word or concept might be chosen. For example, consider the following sub-section from the 'Bribery Act 2010':

R requests, agrees to receive or accepts a financial or other advantage

[11] The only examples of this type are where individual fragments of a compressed provision are not capable of being compressed.

This provision describes the circumstances in which "R" might commit bribery. The relevant actions, "requests", "agrees to receive" and "accepts" are, in this context, closely related. However, they are not synonyms – a person might request something without ever receiving it and, from a rule-drafting perspective, it is important that all of the words are present. For the purpose of a summary, however, we suggest that the following is a reasonable compression:

> *R ~~requests, agrees to receive or~~ accepts a financial or other advantage*

"Accepts" is retained while "requests" and "agrees to receive" are deleted because, intuitively, "accepts" has both the broadest meaning and is the action most likely to be relevant in this context. However, this is a subjective judgment and others may argue a different choice is more appropriate.

Whilst this is a reasonably straightforward compression, it does make the provision syntactically simpler. Interestingly, using the spaCy parser, the dependency parse of the uncompressed text incorrectly identifies "R" as a compound to "requests", with "requests" as the subject of the verb "agrees". By contrast, the dependency parse of the compressed provision correctly identifies "R" as the subject of "agrees".

3.3.2 Context assumptions

Another consequence of the desire to avoid ambiguity in legal rule drafting is that facts or circumstances which, in non-legal language, would usually be assumed are, in legal rules, made explicit. For example, consider the following sub-section from the 'Children Act 1989':

> *Where a child's father and mother were married to each other at the time of his birth, they shall each have parental responsibility for the child.*

This provision is describing which of a child's parents automatically assume responsibility for the child. The requirement that the father and mother be married "to each other" before they automatically assume responsibility for the child is important from a rule-drafting perspective: a child's parents might be married to other people at the time of his birth and, in that circumstance, different rules should apply. However, we suggest that most readers of this provision will assume

that the words "...father and mother were married at the time of his birth..." *means* "married to each other". We can use this fact to generate the compression:

> *Where a child's father and mother were married ~~to each other~~ at the time of his birth, they shall each have parental responsibility for the child.*

Whilst there is little syntactic difference between the compressed text and the original, we note that the higher frequency of prepositional phrases in legislative language was a key difference from normal language, so removal of these sorts of prepositional phrases may assist overall readability.

3.3.3 Extraneous information

Distinct from the concept of "context assumption" discussed above, we suggest another feature of legal rule drafting is its inclusion of words and concepts which, to most readers, are simply extraneous. That is not to say such words and concepts contribute nothing to the legal rule, but just that most readers will find the information irrelevant (or lack the domain knowledge to understand its purpose). For example, consider the following sub-section from the 'Academies Act 2010':

> *The Academy is to be treated, on the conversion date, as designated by order under section 69(3) of SSFA 1998 as an independent school having that religious character.*

This provision is describing how a religious school should be treated when it is converted into an Academy (an Academy is a type of school which is state-funded but independent from the local authority). We suggest that two compressions might be made to this provision.

The first is an application of "context assumption" discussed above. The provision describes how the Academy is treated "on the conversion date". The inclusion of this language in the provision is important because an Academy might later lose its religious character – it is only on the conversion date that the religious character may be assumed. However, we suggest the distinction is not important for most readers. For most readers the question is just whether a school having a religious character maintains that character on conversion to an Academy – readers assume the position being described relates to the time of conversion.

The second compression relates not to a context assumption, but to the language specifying that the designation under the SSFA 1998 is "by order". This language makes clear that the deemed designation is a designation made in the ordinary way under the SSFA (i.e. "by order") rather than in any other way. However, we suggest that for most readers this specificity is irrelevant: readers without an understanding of how legislation and rules can be made will gain nothing from the stipulation that the designation be made by order. We suggest, therefore, that an appropriate summary of this provision is:

> *The Academy is to be treated~~, on the conversion date,~~ as designated ~~by order~~ under section 69(3) of SSFA 1998 as an independent school having that religious character.*

As with the Bribery Act example above, it is interesting to consider the dependency parses of the original and compressed provision. Both parses are complex but we note that, in the (spaCy) parse of the compression, "school" is treated as the prepositional object of "as" and "having" as an modifier of the noun "school". This is a more natural reading of the provision as compared to the parse of the original provision in which "having" is treated as an adjectival clause modifier of "designated" and "school" as the subject of "having".

Chapter 4

Compression models

4.1 Data preparation

The legislative compressions corpus is split into training, validation and test sets. The training set contains 750 examples, the validation set 100 examples and the test set 150 examples. Examples are allocated to the sets randomly.

The news headline data is made available in JSON format and is extracted using the standard 'json' library for Python. The news headline data is also split into training, validation and test sets. The training set contains 120,000 examples, the validation set 30,000 examples and the test set 50,000 examples.

Before the data can be ingested by the compression models it must first be 'tokenized': converted from raw strings to lists of 'tokens'. In broad terms, each word and punctuation mark in a sentence is a token, but some words might be treated as multiple tokens (e.g. 'don't' becomes two tokens, 'do' and 'n't') depending on the tokenization method used. In the rule-based and LSTM models we use spaCy to tokenize and pre-process the data. The BERT model demands its own bespoke 'WordPiece' tokenization method.

Both the LSTM and BERT models frame the compression task as a sequence-to-sequence problem (see section 2.4.3) where the input sequence is the uncompressed text and the target sequence is a sequence of labels indicating whether each token in the uncompressed text should be deleted or retained. An example is shown in figure 4.1 below, where start-and end-of-sequence labels have also been added, as is standard practice. The labels are derived automatically by comparing the

uncompressed and compressed texts.

Figure 4.1: Example derived labels

Original	<s>	A	company	must	have	articles	of	association	prescribing	regulations	for	the	company	</s>
Compressed	<s>	A	company	must	have	articles	of	association	</s>					
Target labels	2	1	1	1	1	1	1	1	0	0	0	0	0	3

4.1.1 LSTM data preparation

The input to the LSTM model consists of several components: (i) the text of the uncompressed input, tokenzised using spaCy; (ii) a list of the vocabulary used in the text inputs; (iii) word embeddings in respect of the vocabulary; and (iv) the target labels.[1]

When ingesting the input text, the LSTM model 'looks-up' the word embedding for each token in the word embeddings matrix. The word embeddings are taken from Google's pre-trained '100 billion words' embeddings, which has embeddings for about 3 million words and phrases, trained on approximately 100 billion words of the Google News dataset.[2] The embeddings have length 300. As discussed in section 2.5, using pre-trained word embeddings is a type of implicit transfer learning. We choose to use these embeddings because: (i) they are trained on a much greater volume of data than is available to us as training data; and (ii) they are drawn from a similar domain as our (pre) training data (news content). The full embedding data is very large (*c*. 1.5GB raw and *c*. 3.5GB when held in memory) and instead we extract only the embeddings needed for our vocabulary. We do this using the 'Gensim' library for Python.[3]

4.1.2 BERT data preparation

In order to make use of the pre-trained BERT model, our data inputs must conform to those expected by the model. As noted above, one consequence of this is that the text input is tokenized

[1] https://github.com/richardbatstone/legal_language_compression/tree/master/data/LSTM_parsed_data

[2] The dataset and pre-trained embeddings are available here: https://code.google.com/archive/p/word2vec/, accessed July 2019

[3] https://radimrehurek.com/gensim/, accessed July 2019

using the bespoke BERT tokenizer released with the model. Once the text inputs have been tokenzised, labels are automatically derived in the way described above. However, the label scheme is slightly different for BERT, since '0' is already used as the padding token by the model. Instead, we use '1' as the start-of-sequence label, '2' as the retain label, '3' as the delete label and '4' as the end-of-sequence label. Each training example is packaged as a Tensorflow features object and written to a Tensorflow Records File.[4]

4.2 Rules-based model

4.2.1 Design

The rules-based model[5] is composed as a series of syntactic and regular expression rules which act on an input in sequence to generate a compressed output. To be consistent with the statistical models, the rules are solely deletion-based (i.e. they act by removing tokens from the input text, rather than performing lexical substitutions or syntactic re-ordering). The rules are developed by hand through experimentation on the training and validation sets, with a focus on high precision: narrow, precise rules are preferred to broader, less precise rules. The rules (developed in spaCy) are:

1. *removal of text in parentheses*: language within parentheses is, by construction, often parenthetical to the main content of the sentence. Sequences of tokens within parentheses are removed. Single tokens are left, as these are often cross-references to other sections. For example:

   > *References in this Act to a child whose father and mother were, or ~~(as the case may be)~~ were not, married to each other at the time of his birth must be read with section 1 of the Family Law Reform Act 1987 ~~(which extends their meaning)~~.*

[4] https://github.com/richardbatstone/legal_language_compression/tree/master/data/BERT_parsed_data

[5] https://github.com/richardbatstone/legal_language_compression/tree/master/models/Rules_models

2. *removal of syntactic sub-trees having "in respect" or "for the purposes of" as their head*: language under these syntactic heads is often included for specificity and cross-referencing purposes and, generally, can be inferred from context. For example:

   > *An authority or body that falls within subsection (1) is only a "public authority" or "public body" ~~for the purposes of the GDPR~~ when performing a task carried out in the public interest or in the exercise of official authority vested in it.*

3. *removal of text following "a reference to/in"*: similar to the above, this type of language is often included for specificity and cross-referencing purposes and can usually be inferred from context. For example:

   > *~~A reference in this section, section 21 or 22 or an applicable Schedule to~~ an auxiliary aid includes a reference to an auxiliary service.*

4. *removal of common extraneous legal phrases*: certain phrases that frequently appear in legislation but which contribute little to an ordinary reader's understanding of a provision are removed. These include "as the case may be", "to any extent" and "so far as is reasonably practicable". For example:

   > *It shall be the duty of each person who has~~, to any extent,~~ control of premises to which this section applies or of the means of access thereto or egress therefrom or of any plant or substance in such premises to take such measures as it is reasonable for a person in his position to take to ensure~~, so far as is reasonably practicable,~~ that the premises, all means of access thereto or egress therefrom available for use by persons using the premises, and any plant or substance in the premises or~~, as the case may be,~~ provided for use there, is or are safe and without risks to health.*

5. *removal of certain alternative sub-clauses*: sub-clauses conforming to certain patterns are removed. For example, where the first word of the sub-clause is "or" and the final word is

the same as the word immediately preceding the sub-clause. For example:

> *An officer of a company~~, or a person acting on behalf of a company,~~ commits an offence if he uses, or authorises the use of, a seal purporting to be a seal of the company on which its name is not engraved as required by subsection (2)*

6. *removal of certain "doublets"*: legislative language is peppered with "doublets", single concepts expressed as conjunctions of two near synonymous concepts to ensure generality. For example "safety and welfare", or "fit and proper". Doublets are identified by matching "X and Y" patterns and compressing those patterns where the similarity between the word embeddings for X and Y is sufficiently high. The word embeddings are taken from spaCy's in-built 'Glove' vectors.[6] Similarity is measured as cosine similarity. The similarity threshold is set by trial and error on the training and validation set. Even so, this rule is the least precise of the rules described as word embedding similarity does not necessarily imply synonymy. For example, the word embeddings for "mother" and "father" are similar, but they cannot be treated as synonyms in a legislative context.

4.3 Statistical models: LSTMs

4.3.1 Design

We follow the approach set out in Filippova et al. [2015] and implement an encoder–decoder LSTM network to map from input sequences of tokens to output zero-one sequences indicating whether the token be deleted or retained in the compression.[7] During training, the model minimises the cross-entropy loss between the decoder outputs and the target outputs.

We implement the first and most basic version of the models explored in Filippova et al. [2015]. That is, the inputs to the model are just the raw tokens (which are then converted to word embeddings) and we do not add other information (such as concatenating the embeddings for the input

[6] Glove vectors are an alternative to the Word2Vec vectors discussed in section 2.5. See https://nlp.stanford.edu/projects/glove/, accessed July 2019

[7] https://github.com/richardbatstone/legal_language_compression/tree/master/models/LSTM_models

token with the embedding for the token's syntactic head). This is because we are applying the model to a domain where: (i) we have no way of assessing the accuracy of automatic dependency parsers, though we might reasonably expect them to perform poorly compared to normal language (see, for example, the parsing errors discussed in section 3.3); and (ii) our inputs are not necessarily syntactically complete sentences, which might introduce more noise into a dependency parse.

The 'LSTM base' model reported in section 4.5 has three layers in both the encoder and decoder. Each layer has 128 nodes, using the Rectified Linear Unit activation function. The final layer of the model is a fully connected softmax layer that maps from the decoder output to the target label space.

We do not adopt the strategy in Filippova et al. [2015] of making the model 'bi-directional'. In a bi-directional model, two models are trained in parallel with the second model being fed the inputs in reverse order. The outputs of the two encoders are then concatenated before being passed to the decoder. This approach is intended to assist with the 'vanishing gradient' problem discussed in section 2.4.2. However, experimentation on the training and validation sets did not demonstrate noticeable improvement when using a bi-directional model.

The model's other hyper-parameters are also selected through experimentation on the training and validation sets. The model is trained using the 'ADAM' optimiser and an initial learning rate of 0.002. The ADAM (Adaptive Moment Estimation, (Bengio and LeCun [2015a])) optimiser is a variation of the gradient-descent by back-propagation algorithm described in section 2.4.1. Rather than applying a single learning rate for parameter updates, ADAM calculates parameter specific updates for different parameters. The magnitude of the updates also varies depending on the gradient trend: updates are larger where the gradient is in the same direction over successive updates and lower where the direction oscillates. ADAM seeks to combine two intuitions: (i) updates to parameters which are infrequently updated should be larger than those which are frequently updated; and (ii) updates should be larger where the direction of the gradients is consistent. It has become common practice to use the ADAM optimiser (or variations on it) in deep learning models.

A 'dropout' layer is applied to each layer of the model. 'Dropout' is the process whereby the

contributions of each of a layer's nodes are, during training, ignored with a certain probability. For example, a dropout rate of 0.2 implies that during each training step one fifth of the nodes in that layer will be ignored for the purpose of computing the layer's output (Srivastava et al. [2014]). Dropout is intended to improve the model's ability to generalise. The intuition is that by randomly ignoring the contributions from a sub-set of nodes in a layer, it becomes less likely that the layer "memorises" the training data and, instead, is more likely to learn multiple copies of generally applicable features. The model uses a dropout rate of 0.2 across all layers.

The training strategy is also augmented through 'gradient clipping'. In gradient clipping, the gradients (derived through back propagation) forming the basis of the parameter updates are 'clipped' such that they are subject to a maximum value. This is intended to smooth the training process and prevent the parameters from quickly converging to poor quality local optima. In our model, the gradients are clipped to have a maximum norm of one.

4.3.2 Implementation

The model is built using the 'Tensorflow'[8] library for Python. Filippova et al. [2015] did not release code and instead the model is implemented by simplifying and adapting the code base used in Google's Neural Machine Translation tutorial.[9]

When training the model on the news headline corpus, we decrease training time by deploying the model on Google's cloud-based Colab GPUs.

4.3.3 Transfer learning

We begin by training the model on the news headline corpus of 200,000 parallel sentences and compressions. The model is trained to 25 epochs with a batch size of 25 on the corpus training set.

The word embeddings are held fixed during training. That is, we do not allow the model to update the embeddings to learn specific representations for the deletion task. It is likely that allow-

[8] https://www.tensorflow.org/, accessed July 2019.
[9] https://github.com/tensorflow/nmt, accessed July 2019.

ing the model to update the embeddings would lead to better performance on the news headline corpus, though it is less clear that it would benefit the legislative training task.[10] In one experiment, we try training word2vec embeddings on the legislative corpus and using these as the inputs to the model. However, rather than pursue these experiments with the LSTM model we explore using a general purpose language model (BERT).

In order to establish a base line, we first apply the news headline trained model to the legislative corpus. We then experiment with two approaches to explicit transfer learning:

1. *traditional fine-tuning of the model on the legislative corpus*: in this approach, we take the LSTM base model and train it for a further ten epochs on the legislative training data. We do not adapt the learning rate or procedure from the original training procedure;

2. *language model fine-tuning (ULMFiT)*: in this approach, we follow the procedure (and adopt the suggested hyper-parameters) outlined in Howard and Ruder [2018] for language model fine-tuning. We take the LSTM base model and train it for ten epochs on the legislative training data using the following training procedure:

   (a) linearly increasing the learning rate for the first 10 per cent of training steps to a maximum learning rate of 0.01, before linearly decreasing the learning rate for the remaining 90 per cent of training steps to a minimum of 1/32 of the maximum learning rate (0.0003125);

   (b) decreasing the learning rate for each layer of the model, such that the learning rate of layer l is the learning rate of layer $(l+1)/2.6$; and

   (c) gradually 'unfreezing' the model's layers such that the final layer is trained for the entire training process and successive layers trained only after a set portion of training

[10] Heuristically, there is a trade off between learning task-specific representations and learning representations that generalise well. One reason for using pre-trained embeddings from a large corpus is that it should allow the model to generalise to inputs containing words that did not appear in its training data, provided the training data did contain words that are 'similar' to the unseen words. However, this relies on the relationships which exist between the general corpus of pre-trained word embeddings being maintained. If the embeddings for the training data vocabulary are updated to become task-specific representations during training then these relationships are eroded.

steps. So, given our model has seven layers, the second layer is only trained after 1/7th of the training steps have elapsed, the third layer after 2/7th have elapsed etc.

For comparison, we also train a model solely on the legislative corpus (i.e. without pre-training on the news headline data). As above, we train the model for ten epochs with the same hyper-parameters as used for the LSTM base model.

4.4 Statistical models: BERT

4.4.1 Design

We design an encoder–decoder transformer network in which the encoder is a pre-trained language model, BERT.[11]

We use the BERT-base uncased model which is formed of twelve transformer layers, each with a hidden size of 768 and twelve self-attentive heads. As the name suggests, this model assumes all tokens have been lower-cased. The outputs from the final layer of this model are passed to a decoder transformer. The decoder has six layers, each with a hidden size of six,[12] three self-attentive heads, three heads attending to the encoder outputs and two fully-connected feedforward networks. The feedforward networks in each decoder layer have a hidden size of 512.

Like the LSTM model, at training time the model minimises the cross-entropy loss between the decoder output and the target outputs. The entire model (including the BERT encoder) is trained simultaneously. We follow the training strategy suggested in research notes accompanying the release of the model.[13] We use the ADAM optimiser, augmented with a linear 'warm-up' in which the learning rate is linearly increased to a set maximum over an initial number of training steps. We set the maximum learning rate to be $2e^{-5}$ and a warm-up proportion of 0.1. A dropout rate of 0.1 is applied to each component of the decoder layers.

[11] https://github.com/richardbatstone/legal_language_compression/tree/master/models/BERT_models

[12] Note, our decoder hyper-parameter choices are somewhat constrained by our task set-up: the tranformer hidden layer size is equal to the embedding size and the number of attention heads must evenly divide the embedding size. We artificially increase our embedding size to six (adding an unused label) to be able to apply the model to our task.

[13] https://github.com/google-research/bert#fine-tuning-with-bert, accessed July 2019

4.4.2 Implementation

The model is built using the Tensorflow library for Python. The encoder is imported directly as a 'Tensorflow Hub' model and the decoder built on top of it follows the architecture and code base set out in the Tensorflow transformer model.[14] The model is deployed on Google's cloud-based Colab GPUs and the model data stored in Google Cloud Platform storage 'buckets'.

4.4.3 Transfer learning

As with the LSTM model, we begin by training the model on the news headline corpus. The model is trained to twenty epochs with a batch size of 32 on the corpus training set.

As discussed in section 2.5, adapting a pre-trained language model to a specific task is a form or implicit transfer learning: it is hoped that the variation on the language modelling task and the broad source domain allow the model to capture sufficiently general features of language to make it a useful starting point for our task and our domain. Unlike with the word embeddings used in the LSTM model (which we held fixed during training) we allow the model to update the pre-trained language model parameters during training.

In order to establish a base-line, we first apply the model to the legislative corpus. We then experiment with training two further models:

1. *traditional fine-tuning of the model on the legislative corpus*: we take the BERT base model and train it for a further ten epochs on the legislative training data. As with the procedure for training on the news headline corpus, we adopt a 'warm up' period during which the learning rate is slowly increased to maximum of $2e^{-5}$;

2. *training only on the legislative corpus*: we train a model solely on the legislative corpus (i.e. without pre-training on the news headline data).

[14] https://github.com/tensorflow/models/tree/master/official/transformer, accessed July 2019

4.4.4 BERT-rules model

As a development and as discussed in section 4.5, we also experiment with ensemble models combining the rules-based model and BERT-based models.

Our motivation is to increase the overall performance of the model by combining: (i) the high-precision compressions of the rules-based model; and (ii) the flexibility and generalisability of BERT-based models.

To develop these models we select a sub-set of the legislative compressions corpus for training. In particular, we exclude from this 'targeted' corpus: (i) any training examples which are concatenations of legislative provisions (on the basis that the model will be applied on a sub-section level basis); and (ii) any examples which would be largely compressed by the rules-based model (e.g. compressions that consist only of removing information in parentheses), on the basis that by their removal we reduce the complexity of the compression task that the BERT model must learn. 550 examples in the training set and 76 examples in the validation set are selected as 'targeted' examples. We do not define a 'targeted' test set and instead evaluate the whole model on the original test set.

We experiment with two ensemble models: (i) an ensemble model where we take the BERT model (trained on the news headline data) and train it for a further 10 epochs on the 'targeted' training data; and (ii) an ensemble model where we train the BERT model only on the 'targeted' training data set (i.e. without pre-training on the news headline data). We do not experiment with an LSTM ensemble model because the BERT models are generally more successful.

4.5 Results

4.5.1 Summary

Tables 4.1 to 4.4 show the results for the: (i) LSTM 'base' model (trained only on the news headline corpus); (ii) LSTM 'transfer' model (LSTM base model fine-tuned on the legislative corpus); (iii) LSTM 'ULMFiT' model (LSTM base model fine-tuned according to the ULMFiT procedure); (iv) LSTM 'leg only' model (trained only on the legislative corpus); (v) BERT 'base' model (trained

only on the news headline corpus); (vi) BERT 'transfer' model (BERT base model fine-tuned on the legislative corpus); (vii) BERT 'leg only' model (trained only on the legislative corpus); (viii) 'rules-based' model; and (ix) BERT-rules ensemble models. All evaluations are based on performance on the test set.[15]

In addition to the evaluation metrics[16] described in section 2.3, we also report the 'string match' and 'compression ratio' for each model:

1. *string match* is the proportion of examples for which the model is able to replicate the gold standard. Note, there are six examples in the test set where the original text and the target compression are the same (i.e. no compressions should be made);

2. *compression ratio* is the average ratio of the length of the compressions generated by the model to the length of the uncompressed text. The compression ratio of the original texts to the target compressed texts is 0.63 (i.e. in the 'gold standard' compressions, 37% of tokens are deleted).

Evaluation metric	**LSTM base model**	**LSTM transfer**	**LSTM ULMFiT**	**LSTM leg only**
F1	0.46	0.72	0.72	0.74
String match	0.02	0.01	0.01	0.04
Compression ratio	0.40	0.87	0.88	1.0
ROUGE-2R	35.25	74.93	77.45	87.65
ROUGE-LR	46.35	87.40	89.30	99.97
BLEU	25.06	45.42	46.12	49.85
SARI	34.66	39.26	39.62	38.88

Table 4.1: Results - LSTM models

[15] Model weights are available at https://console.cloud.google.com/storage/browser/rob-bert-legal-compression-public

[16] As noted in section 4.1, the LSTM and BERT models rely on different methods of tokenization. The practical effect of this is that there are 5,937 tokens in the BERT tokenization of the test set whilst there are 6,197 in the LSTM tokenization. Therefore, in order to permit cross-model comparison, we first 'reconstruct' the outputs of the models (i.e. transform the list of tokens into human-readable text). We then tokenize the outputs on a consistent basis (using NLTK) before calculating the evaluation metrics. This approach is not taken for the F1 score and it follows that the F1 score should not be used for fine-grained comparison between the BERT and LSTM models.

Evaluation metric	BERT base	BERT transfer	BERT leg only
F1	0.40	0.55	0.75
String match	0.006	0.01	0.03
Compression ratio	0.36	0.50	0.86
ROUGE-2R	33.98	46.35	72.43
ROUGE-LR	41.37	55.33	84.37
BLEU	24.77	33.26	46.50
SARI	30.22	33.47	30.00

Table 4.2: Results - BERT models

Evaluation metric	Rules based
F1	0.77
String match	0.08
Compression ratio	0.90
ROUGE-2R	83.89
ROUGE-LR	94.52
BLEU	55.92
SARI	36.24

Table 4.3: Results - Rules-based model

Different models compress the input text to different extents and, as discussed in section 2.3, we follow Napoles et al. [2011] in comparing model performance against this factor. The results are discussed in detail below. In summary:

1. *compression ratio of 0.85–0.90*: four models generated compression ratios in this range. Of these models, the rules-based model performed best on all metrics except SARI, on which the LSTM ULMFiT model performed best.

2. *compression ratio of 0.35–0.40*: three models generated compression ratios in this range. Of these models, the BERT-rules ensemble model with transfer learning (results shown in table 4.4) and the LSTM base model performed comparably on the objective metrics. Examining the outputs suggests the BERT-rules ensemble model performed better overall.

Finally, the BERT-rules ensemble model *without* transfer learning generated a compression ratio of 0.72 (results shown in table 4.4). This is the model that, in terms of compression ratio, came closest to the legislative compression corpus (i.e. came closest to the 'gold standard' compression ratio). For this reason, we use this model as the basis for our extrinsic evaluation experiment (section 4.6).

4.5.2 Base models

Examining the results, we can see that both of the 'base' models (the models that were trained only on the news headline corpus) performed poorly on the legislative corpus. The cause of the poor performance can be seen in the compression ratio and ROUGE scores: both models tended to delete too much. That is, the news headline compressions are in general shorter than the legislative compressions and this prevents the model from transferring well to the new domain. The LSTM base model appears to slightly outperform the BERT base model.

4.5.3 Transfer learning models

Each of the transfer learning models generate longer compressions when compared to the base models and this is accompanied by improved performance across all objective metrics. The performance improvement is most pronounced in the LSTM models (the 'transfer' and 'ULMFiT' models). The compression ratio for the LSTM transfer models increases to 0.87 and 0.88 and the high F1 and ROUGE scores can be seen as a product of the models generally learning to retain more information. The BERT transfer model does not improve as significantly, seemingly taking longer to forget its deletion-heavy approach to the news headline data (perhaps as a result of the number of parameters in the model).

Examining the outputs, we can see that the LSTM models tend to have success where the compression can be approximated by deleting the start or end of the input. For example, given the input (and target in strikeout):

A person ~~appointed under paragraph 1(e)~~ may resign from the Committee by giving notice ~~to the Committee~~.

The LSTM transfer model returns:

A person appointed under paragraph 1(e) may resign ~~from the Committee by giving notice to the Committee~~.

And the LSTM ULMFiT model returns the similar:

A person appointed under paragraph 1(e) may resign from the Committee ~~by giving notice to the Committee~~.

Both models are able to identify the 'to the committee' language as being capable of compression but both miss the modifying clause 'appointed under paragraph 1(e)'. By contrast, and notwithstanding its lower performance on the objective measures, in this particular example the BERT transfer model is able to return the better compression:

A person ~~appointed under paragraph 1(e)~~ may resign from the Committee by giving notice to the Committee.

As evidenced in the BERT transfer model's lower compression ratio, it continues to over-delete in many cases. For example, given the input (and target in strikeout):

Where there is an agreement to sell ~~specific~~ goods and ~~subsequently~~ the goods, without any fault ~~on the part~~ of the seller or buyer, perish before the risk passes ~~to the buyer~~, the agreement is avoided.

The BERT transfer model returns:

~~Where~~ there is an agreement to sell specific goods and subsequently the ~~goods, without any fault on the part of the seller or buyer, perish before the risk passes to the buyer, the agreement is~~ avoided.

Aside from being a poor approximation of the target, this compression is also incomprehensible. In these circumstances the outputs of the LSTM transfer and LSTM ULMFiT models are to be preferred: they return the input text unchanged.

The LSTM ULMFiT model slightly outperforms the LSTM transfer model, but the difference is not significant and over half (79 out of 150) of the test outputs are identical.

4.5.4 Legislative corpus only models

In addition to the base and transfer learning models, we also train LSTM and BERT models solely on the legislative corpus (i.e. without first training on the news headline task).

Whilst the LSTM model appears to perform well, the model actually collapses: it fails to delete any tokens in the test set (evidenced by its compression ratio of 1). We speculate that, with only the pre-trained word embeddings as a source of linguistic information, there is too little information in the legislative corpus to enable successful training of the model. Note, however, that these results are a useful lens for interpreting the evaluation metrics. A model with a compression ratio of 1 outperforms the LSTM ULMFiT model on every metrics, except SARI (where the difference is minimal), notwithstanding that it utterly fails at the target task. This highlights the 'recall' bias in the metrics (and why it is only sensible to compare models that generate similar compression ratios).

By contrast, the BERT model trained only on the legislative corpus is able to achieve a compression ratio similar to the LSTM transfer models (0.86). The model has a markedly lower SARI score but more comparable ROUGE scores and a higher BLEU score. Inspecting the outputs, we see that it also behaves similarly to the LSTM transfer models in that it largely performs the compressions by deleting the start or end of the inputs (or else does nothing). For example, given the input (and target in strikeout):

> *in any way increases his liability ~~as at that date to contribute to the company's share capital or otherwise~~ to pay money to the company.*

The model returns:

> *in any way increases his liability as at that date to contribute to the company's ~~share capital or otherwise to pay money to the company.~~*

4.5.5 Rules-based model

The rules-based model has the highest compression ratio, deleting (on average) 10% of the inputs (against the target of 37%). However, its compression ratio is still comparable to the transfer learning statistical models and it outperforms those models on several metrics: (i) it achieves the highest F1, BLEU scores and ROUGE scores, reflecting the highly precise nature of its deletions; and (ii) it achieves the highest proportion of string-for-string reconstructions on the test set (8%). As expected, where the model does compress inputs, the outputs are of a high quality. For example, given the input (and target in strikeout):

> *~~For the purposes of the GDPR,~~ the following ~~(and only the following)~~ are "public authorities" and "public bodies" ~~under the law of the United Kingdom—~~*

The model returns:

> *~~For the purposes of the GDPR,~~ the following ~~(and only the following)~~ are "public authorities" and "public bodies" under the law of the United Kingdom—*

By contrast, the BERT transfer model deletes too much of the start of the sentences whilst the LSTM transfer models do not make enough deletions.

4.5.6 BERT-rules ensemble models

In our final experiment, we see if we can combine the high-precision, low-compression of the rules-based model with the more general BERT compression models (as described in section 4.4.4).

As can be seen, the ensemble model which is pre-trained on the news headline data generates a compression ratio of 0.39. This is similar to the base statistical models and the model shows improved performance over the base models on several metrics.

The ensemble model which is not pre-trained on the news headline data generates a compression ratio of 0.72, which is not directly comparable to the other models but is closest to the 'gold standard' compression ratio of 0.63.

F1 score is an approximation for these models as alignment between the target labels and the predicted labels is lost as a result of using both spaCy and BERT tokenization (and reconstruction).

Evaluation metric	**BERT-rules ensemble (transfer model)**	**BERT-rules ensemble (leg only model)**
F1	*0.42*	*0.57*
String match	0.00	0.006
Compression ratio	0.39	0.72
ROUGE-2R	36.07	63.11
ROUGE-LR	44.80	73.62
BLEU	27.30	45.19
SARI	32.67	37.20

Table 4.4: Results - BERT-rules ensemble models

4.6 Extrinsic evaluation of compressions

As noted in section 2.3, evaluation of the grammatical correctness and informativeness of compression model outputs is typically carried out by humans. For example, Filippova et al. [2015] evaluated their system's output by having three human reviewers each rate 200 example outputs on a five-point Likert scale for the output's *readability* and *informativeness*.

Rather than evaluating the raw model outputs we design an experiment to assess the extrinsic usefulness of the compressions. That is, we embed the compressions in a real-world task and assess performance on that task. This approach is motivated by the broader challenge of making legislation more comprehensible. The task demands a range of tools and we would expect sentence compression to form part of an overall system, rather than being deployed in isolation. Therefore

we attempt to evaluate the model's performance in context.

4.6.1 Experiment design

Our extrinsic evaluation involves embedding the compressions in a question-answer task as follows:

1. we take four extracts from legislation and apply the BERT-rules ensemble model (without pre-training) to generate compressed versions;[17]

2. for each of the four extracts we design five comprehension questions based on the text of the extracts (twenty questions in total);

3. we take an off-the-shelf question-answering model,[18] provide it with the uncompressed extracts and then use it to generate answers to the comprehension questions;

4. we repeat the process with the compressed extracts; and

5. we ask human reviewers to assess (on a five-point Likert scale): (i) how grammatically correct the system response is; and (ii) how well the response answers the question.

The reviewers provide their assessment by way of a questionnaire. Each reviewer assesses ten responses based on the uncompressed extracts and ten responses based on the compressed extracts (the reviewers are not informed of the basis of the responses). Full details of the compressed/uncompressed extracts and the questions are linked to in Appendix A.

4.6.2 Experiment results

Seven reviewers participated in the experiment, three of whom had had some form of legal training. The results are shown in table 4.5 below.

[17] We choose to use the BERT-rules ensemble model without pre-training in the extrinsic evaluation as it generates a compression ratio closest to the target ratio.

[18] We use the open source 'cape' suite of libraries: https://github.com/bloomsburyai/cape-webservices. We run the cape model by deploying its docker image on a google cloud compute engine and make no amendments to the model.

Question	Uncompressed text (average score)	Compressed text (average score)
How grammatically correct is the response?	3.42	3.41
How well does the response answer the question?	3.41	3.39

Table 4.5: Results - Extrinsic evaluation

As can be seen, the results are almost identical for the compressed and uncompressed texts. That is, the evidence of this experiment indicates that: (i) the compressed legislation is no less grammatical than the uncompressed legislation; and (ii) applying the question-answering model to the compressed legislation did not increase the informativeness of the answers as compared to the uncompressed legislation.

We suggest these are interesting initial results but do not place much weight on these conclusions given the small number of assessed examples (twenty) and the inherent difficulty of attributing performance in extrinsic evaluations. For example, it is interesting that the reviewers only rated the grammatical correctness of the uncompressed text responses as 3.42 out of 5: those responses will have been drawn directly from the original (grammatical) legislation extracts, so it is not clear that this measure is giving an evaluation of the grammatical correctness of the underlying text.

Whilst it is interesting to explore extrinsic evaluations of systems we suggest (as noted in section 6.1) replicating the human evaluation carried out by Filippova et al. [2015] would be a valuable exercise for future work.

Chapter 5

Discussion

5.1 Effectiveness of implicit transfer learning

There were two forms of implicit transfer learning in our models: the word2vec embeddings used to form the inputs to the LSTM models and the use of the pre-trained language model weights for the encoder in the BERT models. We saw that there was little difference in the performance of the base models, though we might have expected the BERT base model to perform better given it is intended to capture more complex (and more general) linguistic representations than word2vec embeddings.

However, the BERT model was able to perform comparably well even without pre-training on the news headline corpus, whilst the LSTM model collapsed. This indicates pre-trained language models are generally a stronger starting point than simple word embeddings, even where the language model is derived from a very different domain.

5.2 Effectiveness of explicit transfer learning

The effectiveness of explicit transfer learning (i.e. pre-training on the news headline corpus) is dependent on two factors: (i) whether the pre-training task captures information that is useful for the target task; and (ii) whether that information is effectively transferred during training on the target task.

For the LSTM models, pre-training on the news headline corpus was essential: training on the

legislative corpus alone failed to produce a viable model. We can conclude, therefore, that at least some of the information from the pre-training task was informative for the target task, and at least some of that informative information was transferred.

However, we argue that, in general, our results show that the pre-training task does not capture much information that is useful for the target task. There are two points in support of this:

1. there was little difference between straightforwardly fine-tuning the LSTM model and fine-tuning using the methods described by Howard and Ruder [2018]. If the pre-training task had contained fine-grained, specific information that is useful in the target task we might reasonably expect the ULMFiT method to better transfer this information as compared to straightforward fine-tuning;[1] and

2. the model which generated the compression ratio closest to the target compression ratio was the BERT-rules ensemble model constructed *without* pre-training on the news headline corpus.

We suggest the reason the pre-training task was relatively ineffective for the target task can be seen in our observations about the structural and (corpus level) linguistic differences between legislative language and news article language. From a linguistic perspective, we saw in section 3.1.2 that the FRE score (a measure of readability), the frequency of prepositional phrases and the diversity of the vocabulary are all markedly different in legislative language as compared to news article language. From a structural perspective, we saw in section 3.1.1 that there is not a straightforward equivalence between syntactically complete legislative sentences and news article sentences: the building blocks of legislative language are sub-sections which may or may not be complete sentences.

We discuss what might be done to address these differences (and thereby achieve more effective transfer learning) in section 5.3 below.

[1] Although it is possible that a different choice of hyper-parameters for the transfer learning exercise might have led to better performance. We note, for example, that the hyper-parameters suggested by Howard and Ruder [2018] were intended to fine-tune a language model and not an encoder–decoder sequence-to-sequence model.

5.3 Combining statistical and rules-based methods for simplification in the legal domain

Noting that the pre-training task was relatively ineffective for the target task, it is natural to consider whether we could re-frame either the pre-training task or the target task such that transfer learning (whether implicit or explicit) can be made more effective.

The BERT-rules ensemble models can be seen as a starting point for this type of approach. By selecting a 'simpler' sub-set of the training set and (at test time) pre-processing the inputs by applying the rules-based model, we are effectively augmenting the compression task with the intention of making it more similar to the pre-training task. In the case of the BERT-rules ensemble model *with* pre-training, we are aiming to make the legislative compression task look more like the news headline compression task. In the case of the BERT-rules ensemble model *without* pre-training, we are simply aiming to make the linguistic features of the target task domain (legislative language) more similar to the language on which the underlying BERT language model was trained (Wikipedia and the BooksCorpus). To the extent these augmentations do lead to more similar language domains or tasks, we expect that to enable better transfer learning.

In section 4.5.6 we saw that the BERT-rules ensemble model with pre-training performed well on the objective metrics against the 'base' models (the models to which it has the most similar compression ratio). We also saw that the ensemble model without pre-training generated a compression ratio most similar to the target compression ratio. Examining the models results, we can also see that they are beginning to exhibit some desirable features: suggesting compressions to already semi-compressed texts that were not produced in either the rules-only or statistical-only models. For example, consider the following model outputs:

Model	Output
Original	For the purposes of the GDPR, the following (and only the following) are "public authorities" and "public bodies" under the law of the United Kingdom—
Target	The following are "public authorities" and "public bodies"—
BERT base	following) are "public authorities" and "public bodies" under the
Rules based	The following are "public authorities" and "public bodies" under the law of the United Kingdom—
BERT leg only	For the purposes of the GDPR, the following (and only the following) are "public authorities" and "public bodies" under the law of the United Kingdom—
BERT rules ensemble (both models)	following are "public authorities" and "public bodies"

Table 5.1: Sample model outputs

The BERT-rules ensemble models are the only models to accurately compress both the prepositional phrase "under the law of the United Kingdom" and the parenthetical text. However, both ensemble models do still unhelpfully remove the 'The' from the start of the compression.

Given the lack of parallel data in the legal domain, we suggest that further advancements will need to continue to focus on these ensemble approaches. There are aspects to the compression task that are very difficult to achieve in a rules-based model (e.g. identifying which 'doublets' to compress) and in these instances we would like to bring a statistical model to bear. However, meaningful compressions can be achieved using a rule-based model and we would like to leverage these results unless and until they can be replicated in statistical models.

Chapter 6

Conclusion and further work

6.1 Further work

There are a number of avenues for further work around the simplification of legal text. At a high level, the work in this thesis could be expanded to:

1. develop general simplification models for legislative text (rather than only compression models); and

2. develop compression (or general simplification) models for non-legislative legal text (e.g. contractual or regulatory text).

We suggest more focused work could also be undertaken to better quantify the difference between the compression pre-training task and the target task. In section 5.3 we began to explore augmentation of the target task/domain with some encouraging results. We have yet to explore augmentation of the pre-training task but could consider applying the approaches described in Ruder [2019] to more rigorously select the pre-training and training data. Alternatively, rather than relying on language models pre-trained on non-legal language (i.e. BERT and others), work could be carried out to train a large-scale language model on legal language.

In section 4.6 we explored an initial extrinsic evaluation of the legislative compression models. This evaluation could be expanded (to include more questions / reviewers) and adapted so as to allow direct comparisons with the evaluation carried out by Filippova et al. [2015].

Equally, whilst we have contributed a parallel corpus of legislative compressions, further work could be undertaken to expand that corpus or to develop a general simplification corpus for legal text.

6.2 Conclusion

We have sketched a motivation for developing the novel task of automatic text simplification in the legal domain and explored the key differences between legislative language and ordinary language.

Restricting text simplification to a text compression exercise, we contribute a parallel corpus of legislative compressions and describe their common themes.

We explore and compare numerous methods of applying rules-based and statistical compression models to the low-resource legislative domain. We describe the three highest performing models, each generating encouragingly convincing compressions, at different 'compression ratios':

1. a high-precision rules-based model achieving a compression ratio of 0.90 (F1 0.77, SARI 36.24);

2. a BERT and rules-based ensemble model trained only on the legislative corpus, achieving a compression ratio of 0.72 (the closest to the target compression ratio) (F1 0.57, SARI 37.20); and

3. a BERT and rules-based ensemble model pre-trained on the news headline corpus before being fine-tuned on the legislative corpus, achieving a compression ratio of 0.39 (F1 0.42, SARI 32.67).

We suggest further advancements might be made following the ensemble model approach and suggest topics for future work.

Bibliography

Y. Bengio and Y. LeCun, editors. 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings, 2013. URL https://openreview.net/group?id=ICLR.cc/2013.

Y. Bengio and Y. LeCun, editors. 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015a. URL https://iclr.cc/archive/www/doku.php%3Fid=iclr2015:accepted-main.html.

Y. Bengio and Y. LeCun, editors. 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015b. URL https://iclr.cc/archive/www/doku.php%3Fid=iclr2015:accepted-main.html.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2):157–166, March 1994. ISSN 1045-9227. doi: 10.1109/72.279181.

S. Chopra, M. Auli, and A. M. Rush. Abstractive sentence summarization with attentive recurrent neural networks. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 93–98, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1012. URL https://www.aclweb.org/anthology/N16-1012.

M. Curtotti and E. C. McCreath. A corpus of australian contract language: Description, profiling and analysis. In Proceedings of the 13th International Conference on Artificial Intelligence and Law, ICAIL '11, pages 199–208, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0755-0. doi: 10.1145/2018358.2018387. URL http://doi.acm.org/10.1145/2018358.2018387.

M. Curtotti and E. C. McCreath. A right to access implies a right to know: An open online platform for research on the readability of law. Journal of Open Access to Law, vol. 1, no.1, 2013.

J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. CoRR, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

K. Filippova, E. Alfonseca, C. Colmenares, L. Kaiser, and O. Vinyals. Sentence compression by deletion with lstms. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP'15), 2015. URL https://www.aclweb.org/anthology/papers/D/D15/D15-1042/.

J. R. Firth. A synopsis of linguistic theory 1930-55. 1952-59:1–32, 1957.

R. Flesch. A new readability yardstick. Journal of Applied Psychology, 32(3):p221 – 233, June 1948. ISSN 0021-9010. URL http://libezproxy.open.ac.uk/login?url=http://search.ebscohost.com.libezproxy.open.ac.uk/login.aspx?direct=true&db=pdh&AN=apl-32-3-221&site=ehost-live&scope=site.

S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural Comput., 9(8):1735–1780, Nov. 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

J. Howard and S. Ruder. Fine-tuned language models for text classification. CoRR, abs/1801.06146, 2018. URL http://arxiv.org/abs/1801.06146.

K. Knight and D. Marcu. Statistics-based summarization - step one: Sentence compression. In Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, pages 703–710. AAAI Press, 2000. ISBN 0-262-51112-6. URL http://dl.acm.org/citation.cfm?id=647288.721086.

C.-Y. Lin. ROUGE: A package for automatic evaluation of summaries. In Text Summarization Branches Out, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W04-1013.

R. McDonald. Discriminative sentence compression with soft syntactic evidence. In 11th Conference of the European Chapter of the Association for Computational Linguistics, Trento, Italy, Apr. 2006. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/E06-1038.

C. Napoles, B. Van Durme, and C. Callison-Burch. Evaluating sentence compression: Pitfalls and suggested remedies. In Proceedings of the Workshop on Monolingual Text-To-Text Generation, pages 91–97, Portland, Oregon, June 2011. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W11-1611.

C. O. Office of the Parliamentary Counsel. When laws become too complex, 2013.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: A method for automatic evaluation of machine translation. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://doi.org/10.3115/1073083.1073135.

E. Pitler. Methods for sentence compression, 2010.

S. Ruder. Neural Transfer Learning for Natural Language Processing. PhD thesis, National University of Ireland, Galway, 2019.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning Representations by Back-propagating Errors. Nature, 323(6088):533–536, 1986. doi: 10.1038/323533a0. URL http://www.nature.com/articles/323533a0.

H. Saggion, S. Štajner, S. Bott, S. Mille, L. Rello, and B. Drndarevic. Making it simplext: Implementation and evaluation of a text simplification system for spanish. ACM Trans. Access. Comput., 6(4):14:1–14:36, May 2015. ISSN 1936-7228. doi: 10.1145/2738046. URL http://doi.acm.org/10.1145/2738046.

M. Shardlow. A survey of automated text simplification. International Journal of Advanced Computer Science and Applications(IJACSA), Special Issue on Natural Language Processing 2014, 4(1), 2014. doi: 10.14569/SpecialIssue.2014.040109. URL http://dx.doi.org/10.14569/SpecialIssue.2014.040109.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res., 15(1):1929–1958, Jan. 2014. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=2627435.2670313.

H. Sun and M. Zhou. Joint learning of a dual smt system for paraphrase generation. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2, ACL '12, pages 38–42, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=2390665.2390675.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. CoRR, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

G. Venturi. Parsing legal texts. a contrastive study with a view to knowledge management applications. In Proceedings of the LREC 2008 Ist Workshop on "Semantic Processing of Legal Texts", pages 1–10, 2008.

W. Xu, C. Callison-Burch, and C. Napoles. Problems in current text simplification research: New data can help. Transactions of the Association for Computational Linguistics, 3: 283–297, 2015. URL http://www.cis.upenn.edu/~ccb/publications/publications/new-data-for-text-simplification.pdf.

W. Xu, C. Napoles, E. Pavlick, Q. Chen, and C. Callison-Burch. Optimizing statistical machine translation for text simplification. Transactions of the Association for Computational Linguistics, 4:401–415, 2016. URL https://cocoxu.github.io/publications/tacl2016-smt-simplification.pdf.

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), pages 1353–1361, Beijing, China, Aug. 2010. Coling 2010 Organizing Committee. URL https://www.aclweb.org/anthology/C10-1152.

Appendix A

Code and data list

All code and data is made publically available. The file paths below refer to the GitHub repository for this thesis (and the zip file forming a separate submission): https://github.com/richardbatstone/legal_language_compression. The model files are too large to host on GitHub (or submit) and are instead made available in a Google Storage Bucket (see links below).

data/

data/raw_data

File	Description
data_split.ipynb	Script for splitting the raw (legislative) data into training, validation and test sets
leg_targetted_train_data.pickle	'Targeted' sub-set of training data
leg_targetted_val_data.pickle	'Targeted' sub-set of validation data
leg_test_data.pickle	Test data
leg_train_data.pickle	Training data
leg_val_data.pickle	Validation data
legislative_compressions_tsv.txt	Raw corpus

data/BERT_parsed_data

File	Description
BERT_data_prep.ipynb	Script for preparing data for BERT input
leg_targetted_train_data_128_TFR	BERT input, 'Targeted' sub-set of legislative training data
leg_targetted_val_data_128_TFR	BERT input, 'Targeted' sub-set of legislative validation data
leg_test_data_128_TFR	BERT input, legislative test data
leg_train_data_128_TFR	BERT input, legislative training data
leg_val_data_128_TFR	BERT input, legislative validation data

data/LSTM_parsed_data

File	Description
LSTM_data_prep.ipynb	Script for preparing data for LSTM input
leg_embeddings*	Vocabulary and associated word2vec embeddings for the legislative input text
leg_test*	Legislative test data: labels, original text, spaCy parsed text and vocabulary
leg_train*	Legislative training data: labels, original text, spaCy parsed text and vocabulary
leg_train_targetted*	'Targeted' sub-set of legislative training data: labels, original text, spaCy parsed text and vocabulary
leg_val*	Legislative validation data: labels, original text, spaCy parsed text and vocabulary
leg_val_targetted*	'Targeted' sub-set of legislative validation data: labels, original text, spaCy parsed text and vocabulary
total_vocab	Total legislative dataset vocabulary

models/

models/LSTM_models

The LSTM models are quite large (each in excess of 100MB) and instead of hosting them on GitHub, they are hosted in a publically accessible Google Storage Bucket:

File	Link
LSTM_base	https://console.cloud.google.com/storage/browser/rob-bert-legal-compression-public/public/LSTM_base
LSTM_leg_only	https://console.cloud.google.com/storage/browser/rob-bert-legal-compression-public/public/LSTM_leg_only
LSTM_transfer	https://console.cloud.google.com/storage/browser/rob-bert-legal-compression-public/public/LSTM_transfer
LSTM_ULMFiT	https://console.cloud.google.com/storage/browser/rob-bert-legal-compression-public/public/LSTM_ulmfit

File	Description
model_test_script.ipynb	Script for testing the LSTM models
model_train_script.ipynb	Script for training the LSTM models
model_ulmfit_script.ipynb	Script imlpementing transfer learning using the ULMFiT procedures

models/Rules_models

File	Description
BERT_prep.py	A helper module for preparing BERT inputs as part of the BERT_rules_ensemble model
rule_transformations.py	The rule transformations implemented in the rules based model
rules_BERT_script.ipynb	A helper script for preparing BERT inputs as part of the BERT_rules_ensemble model
rules_only_script.ipynb	Script for running the rules based model

models/BERT_models

The BERT models are quite large (each in excess of 1GB) and instead of hosting them on GitHub, they are hosted in a publically accessible Google Storage Bucket:

File	Link
BERT_base	https://console.cloud.google.com/storage/browser/rob-bert-legal-compression-public/public/bert_base
BERT_transfer	https://console.cloud.google.com/storage/browser/rob-bert-legal-compression-public/public/bert_transfer
BERT_leg_only	https://console.cloud.google.com/storage/browser/rob-bert-legal-compression-public/public/bert_leg_only
BERT_rules	https://console.cloud.google.com/storage/browser/rob-bert-legal-compression-public/public/bert_rules
BERT_rules_leg_only	https://console.cloud.google.com/storage/browser/rob-bert-legal-compression-public/public/bert_rules_leg_only

File	Description
model_run_script.ipynb	Script for training and testing the BERT models (designed to be run on Google Colaboratory)
beam_search.py	Beam search module for BERT models
model.py	Main decoder transformer model

evaluation/

File	Description
BERT_reconstruct.py	A module to reconstruct the outputs from the BERT models
BLEU.ipynb	Script for calculating the BLEU evaluation metric
LSTM_reconstruct.py	A module to reconstruct the outputs from the LSTM models
PrecisionRecall.ipynb	Script for calculating precision, recall, F1, compression ratio and string-for-string match
ROUGE.ipynb	Script for calculating the ROUGE evaluation metric
SARI.ipynb	Script for calculating the SARI evaluation metric
SARI.py	Underlying module for SARI calculation

evaluation/results

File	Description
BERT_base.pickle	Test results for the BERT base model
BERT_leg_only.pickle	Test results for the BERT leg only model
BERT_rules.pickle	Test results for the BERT rules ensemble model (with pre-training)
BERT_rules_leg_only.pickle	Test results for the BERT rules ensemble model (without pre-training)
BERT_transfer.pickle	Test results for the BERT transfer model
LSTM_base.pickle	Test results for the LSTM base model
LSTM_leg_only.pickle	Test results for the LSTM leg only model
LSTM_transfer.pickle	Test results for the LSTM transfer model
LSTM_ulmfit.pickle	Test results for the LSTM ULMFiT model
Rules.pickle	Test results for the rules based model

evaluation/extrinsic

File	Description
compressed_legislation.docx	Compressed legislation generated by 'BERT_rules_leg_only model' for extrinsic evaluation task
QA_responses.docx	The responses to the comprehension questions generated by the 'cape' QA model for both compressed and uncompressed legislation
uncompressed_legislation.docx	Uncompressed legislation for extrinsic evaluation task

The original surveys can be accessed here:

File	Link
Survey 1	https://docs.google.com/forms/d/e/1FAIpQLSceEYbnpHR3K8y1PVMJ3dxlTwvOsGIE2hQN0hchbr-QC34xLw/viewform?usp=sf_link
Survey 2	https://docs.google.com/forms/d/e/1FAIpQLSfZG7DKNoa1-A_ZNo6g6MIiHOiucf1nurWwAy2DJs_UQe5qwg/viewform?usp=sf_link