Project Report: WikiHow

Granit Mullahasani Dula
granit.dula.17@ucl.ac.uk

Klajdi Lamce
klajdi.lamce.16@ucl.ac.uk

Nikolay Bortsov
nikolay.bortsov.17@ucl.ac.uk

Alexandros Frangos
alexandros.frangos.17@ucl.ac.uk

Abstract

This paper compares three current state of the art abstractive text summarisation models: Generative Adversarial Nets (GANs), Sequence to Sequence (seq2sec) model with a single Attention Mechanism using Gated Recurrent Neural Networks and a Sequence to Sequence model using Transformers. The three models are compared for the task of summarising WikiHow articles. The models are all fed with data pre-processed in the exact same way and all make use of the GloVe (Global Vectors for Word Representation) (Pennington et al., 2014) embedding representation, for fair comparison of results and for more optimal performance as a result of these data processing strategies. [COMMENT: Add more here such as results, which found to be the best performing.]

1 Introduction

In the big data era, there has been a dramatic increase in the amount of text data from a variety of sources, making automatic text summarisation, an important tool in a lot of processes. Important applications of text summarisation in Natural Language Processing related tasks include news summarisation, legal documents summarisation and headline generation. Automatic text summarisation involves generating a shorter version of a piece of a text while preserving key information content and overall meaning. The two main methods of summarisation are extraction and abstraction. The former, extracts sentences from the original text to create a summary, whereas the latter, learns an internal language representation to generate more human-like summaries, paraphrasing the intent of the original text. On this paper, abstractive text summarisation models are built which generate headline styled summaries.

Most existing text-summarisation datasets consist of news articles which follow a specific style, therefore models trained on only news dataset might not generalise well to other tasks. On this paper we use the WikiHow dataset (Koupaee and Wang, 2018) which consists of more than 230,000 long-sequence pairs, which includes a wide variety of articles written in different styles, overcoming the limitation that news datasets impose.

We use GloVe for the vector space representation of words, which combines the advantages of the two major model families for learning word vectors: global matrix factorization and local context window methods. GloVe makes use of global statistics (word co-occurrence) to obtain word vectors instead of just relying on local statistics (local context information of words) (Pennington et al., 2014).

In this work, we use sequence-to-sequence (seq2seq) based models all of which have an encoder-decoder architecture. They all take in an input sentence and output sentence as input into the encoder and decoder respectively. The first of these types of models considered is the attention-based model which consists of an encoder, decoder and an attention layer (Mnih et al., 2014) using Gated Recurrent Neural Networks (GRU) (Cho et al., 2014) for the encoder and decoder. The attention mechanism overcomes the problem commonly suffered by standard RNN models, which is vanishing gradients caused by backpropagation of long sequences. This has the effect of losing information on distance words in a sequence. the attention layer solves this by focusing only at specific parts of the sequence only, by assigning weights on its layer that are larger on the more focused regions of the sentence.

Another type of network used in this paper is Generative Adversarial Network (GAN). GAN utilises 2 models in parallel: the generator model that generates the summaries for a given text and the discriminator that judges how good the generated summery is. These to models train themselves against each other, where generator tries to fool the discriminator and the discriminators tries to catch the summaries generated by the generator. Once the generator is able to generate summaries that cannot be distinguished from that written by people by the discriminator the model is sufficient.

Finally, the last model considered is a transformer (Vaswani et al., 2017) model. This architecture is very well known in the natural language processing community and is core to many of the state-of-the-art models today such as GPT-3 (Brown et al., 2020) and BERT (Devlin et al., 2018). The transformer model has the advantage of parallelisation of the input data, because unlike the attention based model, the data is fed in all at once in parallel, rather than token by token through the entire sequence. As a result training times improve greatly. It also happens to perform really well, as a result of its architecture. Each of the models are covered in more detail in section 3.1 Models.

We make use of ROUGE scores to evaluate the performance of the models built. [COMMENT: Add final results here of all three models.]

2 Related Work

2.1 A Neural Attention Model for Sentence Summarization

Neural Networks were first used for abstractive text summarisation in 2015, with the purpose of generating summary words utilizing a local attention-based model by conditioning it on the input sequences. In this work a Bag-of-Words encoder, a Convolutional Encoder and an Attention-Based Encoder were used. The Bag-of-Words model was used as a base model and was used to distinguish between content words from stop words or embellishments. This model however, fails to represent well contiguous phrases. To overcome the limitations of the Bag-of-Words model, the paper proposes a convolutional encoder which utilizes a standard-time-delay neural network (TDNN) architecture, which alternates between max pooling layers and temporal convolution layers. This, therefore, allows local interactions between words without also requiring the context of words. Even though the Convolutional Encoder architecture improves on the Bag-of-Words model, it can only produce a single representation of the input. This limitation was overcome by using an Attention-Based Encoder. The attention-based encoder was utilised to exploit the learned soft alignment to weight the input based on the context to construct a representation of the output.

2.2 Abstractive Text Summarisation using Sequence-to-sequence RNNs and Beyond

In this work, abstractive text summarisation is modeled using Encoder-Decoder Recurrent Neural Networks (RNN) with Attention. The encoder consists of a bidirectional GRU-RNN (Chung et al., 2014) and the decoder consists of a unidirectional GRU-RNN. The authors also proposed several novel methods to address critical problems in abstractive text summarisation such as capturing the hierarchy of sentence-to-word structure and modeling key-words. Instead of just using word-embeddings for the representation of the input document, additional linguistic features such as Term Frequency (TF) and Inverse Document Frequency (IDF) were incorporated to capture keywords. *ADD description of modelling unseen words using switching generator-pointer, and capturing hierarchical document structure with hierarchical attention.

2.3 Abstractive Text Summarization using Attentive Sequence-to-Sequence RNNs:

This paper expands the work done by (add authors from above paper) and aims to explore the effectiveness of different methods for attention. The authors implemented three mechanisms for scoring functions: global attention using the dot product scoring function, bilinear form, and a new format in which an output projection from the hidden states of the RNN encoder to scalar is used for the scoring function. In this work, three models were employed: the first model applied unidirectional LSTM in both the encoder and the decoder; the second model was implemented using bidirectional LSTM in the encoder and unidirectional LSTM in the decoder; and the third model utilised a bidirectional LSTM encoder and an LSTM decoder with global attention. ADD MORE.

[COMMENT: Add one paragraph for GANS]

[COMMENT: Add one paragraph for Transformer based model] [COMMENT: Add one paragraph for word embeddings.] [COMMENT: Add references..]

3 Methods

3.1 Models

3.1.1 Sequence-to-sequence (seq2seq) models

Two of the three models which are evaluated in this paper are variations on sequence-to-sequence (seq2seq) models. Seq2seq models are primarily used in the field of natural language processing, as they can take a sequence of a particular length as input and produce a sequence of a potentially different length as output. This is a useful feature for many tasks, and even a necessary one for some tasks such as machine translation, chatbots, and text summarisation, the machine learning task with which this paper is concerned.

A seq2seq model is defined primarily by its two main components: the encoder and decoder. Broadly speaking, the encoder is fed an input sequence and produces some internal representation, which the decoder then uses to produce an output sequence. The two seq2seq models evaluated in this paper primarily differ in what these two components (encoder and decoder) are comprised of. Our two seq2seq models are fully defined below.

3.1.2 Seq2seq model with Attention

To start with, the encoder of the first seq2seq model contains an embedding layer, which is the mapping of the token inputs into vector representations, using a pre-prepared weights matrix built based on the GloVe embeddings. It also contains a single layer, singular GRU unit which takes in the result of the embedding layers and produces an output for the encoding. This output represents contextual information about the sentence. As a result, this contextual data, in form of a single vector representation, is passed to the decoder layer in the architecture, as the output state and hidden state.

For the decoder layer, it is also comprised of the embedding layer for the vector representation of the input sequence. However, there is an additional dropout layer that regularises the model by hiding some of the data in the embedding. Then there is the attention layer which is essentially a linear layer that learns weights which learn the strength of connection between the embedding input of the decoder, and its hidden state, which comes from the encoder's output. This then gets applied to the encoder's output via a batch matrix-matrix product and the result of this as well as the decoder's embedding input, gets passed through another attention layer, to learn the connections between the input of the decoder and the output of the encoder (the context). This then gets passed through the ReLU activation layer and the result of this, as well as the hidden state, is passed through another GRU unit. This unit helps learn information about the ordering of the words in the input of the decoder and the input of the encoder (via the output of the encoder). Then it finally produces a hidden state output, if needed for further layers, as well as an output state which gets passed through a softmax layer to produce the distribution of possible tokens that could be predicted by the model, to predict the sentence during sentence evaluation.

3.1.3 Seq2seq Transformer model

The transformer model also has an encoder and decoder layer like the attention based model, however, in our case, we chose to use 2 layers for the encoder and decoder each, in order to increase the complexity of understanding that the transformer can attain. We could not increase the number of layers due to limited computational resources. To start with, the transformer has an embedding layer which it receives as input in batches of 4. This then gets added to the positional encoding which creates its own matrix based on the sinusoidal function described in detail in (Vaswani et al., 2017). Essentially, this acts as a representation of the ordering of the words in the sequence for the model to consider, since the model itself does not contain any RNN units in its architecture to do this and so relies on this positional encoding for this process. This as a result allows the transformer to train its input data in parallel unlike models that use RNN units and as a result can train more quickly.

After applying the positional encoding to the input embedding, the result is passed to the encoder. The first layer in the encoder is the multi-head attention layer. This takes the incoming input and applies weights to each of the elements in the input in such a way, that it represents the relation between the words within the input sentence itself. This then gets passed through to be added to the original input itself and passes through a layer normalisation. This helps with gradient flow during backpropagation to help speed up learning. This result then gets passed through a linear layer to add an additional layer of complexity to the understanding of the encoder and the result is then again added to the previous result (before applying the linear layer) and gets normalised once again to produce the ”output” of the encoder. This output is passed to one of the layers in the decoder, which is involved a bit after.

Next comes the decoder part of the transformer. But just before that, it has an initial input, which is the target sequence that the source sequence (which had been input in the encoder part) should be mapped to. This target sequence, just like the source sequence, also gets passed through an embedding layer and has a positional encoding added to its result, for measuring order of words in the target sequence. Then the result of this is passed to a masked multi-head attention layer. This is similar to the multi-head attention layer, with the exception that it has a masking process that is done to remove the affect of the weights for words that are in the future of the sentence, for the word that is being considered. This is to encourage the model to learn to generate words based only on the information it currently has, since the model will be trying to predict the next words of the target sequence when given an actual input to evaluate. So it shouldn’t have access to the weights of future words, hence why they are masked. This is done using a upper triangle matrix with negative infinity in that region of its elements. This is added to the weight matrix which essentially removes the weights for each of the words and their respective future words. This result is then passed through another ”add and norm” layer for better gradient flow in the decoder. Then this result is passed through another normal multi-head attention layer, along with the output of the encoder, through the query, key and value inputs of the layer, mentioned in the paper by Vaswani et al. This allows the model to learn links between the source sequence and the target sequence, through the weights. The result of this, as in the encoder, also gets passed through another layer normalisation and then through a linear layer and again a layer normalisation to produce the output of the decoder. The transformer then passes this result through a final linear to change the shape of the result that is the expected size of the output sequence. It then finally passes through a softmax layer to produce the output probabilities of each of the words in the embeddings, for every word in the output sentence. The word with the highest probability is the word used for that particular word in the output sentence. This can generate predicted sentence. Of course, the model then backpropagates after calculating a loss during training.

3.1.4 Generative Adverserial Network (GAN) model

The GAN model consists of two parts: the Generator and the Discriminator. In an overview the generator generates the wanted output and the discriminator judges it and attempts to distinguish the generator made outputs from the real values. The Generator tries to trick the discriminator and the discriminator tries to catch the generator. Both part of GAN train each other alternating.

The GAN model proposed uses Seq2Seq model with Attention, described above, as the Generator. The model was reused as the generator will be generating the summaries, however, it will not be trained in the same way.

CNN text classification was used as the discriminator. It consists of 3 convolution layers and one linear layer and a linear layer to get the output, probability that if the summary is real or not. The classifier is trained to distinguish between fake generated headlines and the real ones.

Initially the discriminator is pre-trained on fake generated data and real headlines to ensure that discriminator has some previous knowledge about the headlines. After that both models are trained alternating, n number of iterations for discriminator and m number of iterations for discriminator. For discriminator the gradients of cross enterpies between the real prediction and the the target (real) and fake prediction and the target (fake). The real target is set to one and fake target is set to 0. This is used to optimise the discriminator. The generator is trained based on the outcome of the cross entrepy between the discriminator outcome and the target (fake).

4 Experiments

Dataset - The dataset we used was the WikiHow dataset[1]. The repository contained two different links for two versions of the dataset in files called wikihowAll.csv and wikihowSep.csv. The wikihowAll.csv file contains data such that the input

[1]https://github.com/mahnazkoupaee/WikiHow-Dataset

text is the full article with all its paragraphs concatenated. The headline (or the summary of that text) is the concatenation of all summary sentences for each of the paragraphs. Whereas, the wikihowSep.csv file contains data such that the text is a single paragraph and the headline is a single summary sentence for that paragraph. Our models are trained under the wikihowSep.csv dataset, because the focus was on training a model which works with input sentences that are reasonable in size and not too large (and therefore difficult and time consuming to train).

Pre-processing - When the data is loaded a lot of pre-processing is done beforehand, in order to make the training of the model more effective in terms of how quickly it learns the key information about the text and headlines. One of the first steps of pre-processing for both the text and headline was to remove capitalisation and symbols or punctuation. This is to remove the need for the model to learn insignificant information about the grammar of the sentence instead, focus on the semantic meaning of the text. The next step was to remove contracted words for both the text and headline. This changes words like ”wouldn't” to two separate tokens, ”would” ”not”. This makes it easier for the model to understand sentences because it will already have learnt words like ”not” from prior sentences and so does not need to also learn additional information such as how apostrophes work in different words. The next step, which is only applied to the text, is the removal of stopwords. This removes words which do not add much meaning to the sentence. This helps simplify the learning process further for the model since it will not have to learn as much connections between the not very necessary words, which also happen to be very common in language. After this, the rows in the dataset which have headlines with more words than the actual text are removed from the dataset, since the purpose of the model is to summarise and this would conflict with the model's training. Then finally a filter process at the end takes place. This includes converting Unicode characters into their ASCII equivalent. It also includes setting a limit to the size of the text and headline. This is so the model can be trained reasonably enough for a particular text size that is not too large. Its also to accomodate for the input taken by the transformer which is of a fixed size. A maximum text size of 100 was chosen with a maximum summary size of 30. This led to a cut down of only 3.41% of the total amount of data, which is not much loss in data.

Additional Tokens - These added tokens include sos, eos, <pad> and <unk> (for transformer). The sos token is used to indicate the ”start of sentence” token. Similarly, the eos token represents the ”end of sentence” token. The sos is added to the headline, since the model will need to recognise how to generate the next token after sos. Whereas this is not needed for the actual text. The eos token is used in both, since the model needs to know when the sentence finishes for both the text and headline. The <pad> token is used for both the headline and text when training the transformer only. This is because the transformer expects an embedding input of a fixed size and the padding tokens act as filler tokens for sentences that are not as long as the maximum specified length, which is the length of input the transformer expects. Finally, the <unk> token is used to represent words which have not been observed in the dataset, so that it can also represent those words as vector embeddings.

GloVe Embedding - The GloVe embedding is used for the vector representation of the words in the dataset. These vectors relate with each other in a similar manner to the meanings of the words, in the context of the vector space. The GloVe embedding we use is the one trained under the 2014 Wikipedia dataset, with the 200 dimensional vector representation and 50 dimensional vector representation. The fact that it was trained under a Wikipedia dataset makes it useful in the case of this project which is training models under the WikiHow dataset. A 200 dimensional vector was chosen for the attention based model because its large enough to have strong contextual representation, but also not too large to not be able to store in the VRAM of the GPU, when training this model. However, for the transformer model, it was necessary to reduce the dimension size to 50 in order to have enough VRAM to train the model.

Training Strategy - Firstly the data is split into three groups, the training, validation and test set. The data is split with a 6:1:3 ratio respectively, in order of importance between the three groups, where the training set is the most important, then the test set, then the validation set. This results in a training set with 740682 pairs of texts and headlines. The validation set has 123448 and the test set has 370342. The models are trained under the training set and their hyperparameters are tuned based on the loss on the validation set. These hyperparameters are tuned using a grid search approach to find the best combination of performing hyperparameters from a select few. Some of these hyperparameters include the learning rate, the batch size, the dropout and the teacher force rate and others for the transformer models such as the layer size. The models are trained using the ADAMW optimiser (Loshchilov and Hutter, 2017) which is shown by this paper to perform better than ADAM (Kingma and Ba, 2014) which was the previous state-of-the-art for optimisation methods. Additionally, a further attempt was made to improve the learning process of the models, which includes having a learning rate scheduler. This would reduce the learning rate of the model during training, after a certain number of iterations in the training. The purpose was to minimise the step of the gradient descent as it approaches the local minima, to prevent overshooting. However, it was found to have almost no affect, especially since ADAMW was being used as the optimiser, which already has a complex algorithm for handling the optimisation process effectively. The models were trained until time permitted and after that, their losses are compared as well as the evaluation metrics which are explained next.

4.1 Evaluation metrics

To evaluate the performance of the models studied, ROUGE scores were used. In particular, the F1 score for ROUGE-1, ROUGE-2, and ROUGE-L was computed in order to compare with the results presented by (Koupaee and Wang, 2018).

Given a particular true summary (label) and predicted summary (model output) pair, X be the number of overlapping unigrams (i.e. the number of unigrams which are both in the true summary and the predicted summary). The precision and recall for ROUGE-1 is then defined as follows:

- Precision:

$$\frac{X}{|predicted\ summary|} \tag{1}$$

Where $|predicted\ summary|$ is the number of unigrams in the predicted summary (the model output).

- Recall:

$$\frac{X}{|true\ summary|} \tag{2}$$

Where $|true\ summary|$ is the number of unigrams in the true summary (the label).

The F1 score for ROUGE-1 can then be computed from these two values and is equal to

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{3}$$

The F1 score for ROUGE-2 is similarly defined but with bigrams instead of unigrams (i.e. X is the number of overlapping bigrams and $|predicted\ summary|$ and $|true\ summary|$ are the number of bigrams in the predicted and true summaries respectively).

ROUGE-L on the other hand measures overlap in terms of the *longest common subsequence* between the two summaries. Let LCS be the length of the longest common subsequence. Following (Lin and Och, 2004), precision and recall for ROUGE-L are defined as follows:

- Precision:

$$\frac{LCS}{|predicted\ summary|} \tag{4}$$

Where $|predicted\ summary|$ is the number of unigrams in the predicted summary (the model output).

- Recall:

$$\frac{LCS}{|true\ summary|} \tag{5}$$

Where $|true\ summary|$ is the number of unigrams in the true summary (the label).

F1 score for ROUGE-L is again defined as in equation 3 using these values.

5 Results

The results shown will try to both measure and analyse the performance of the attention and transformer based models and demonstrate their predictive capabilities with comparable example inputs between the two and their corresponding outputs. More specifically, the loss curves of each of the models will be compared as well as their ROUGE metric scores, which will be a mean score across the entire test set which has 370,342 sentence and summary pairs. Below is a screenshot of the loss curves for each of the models.

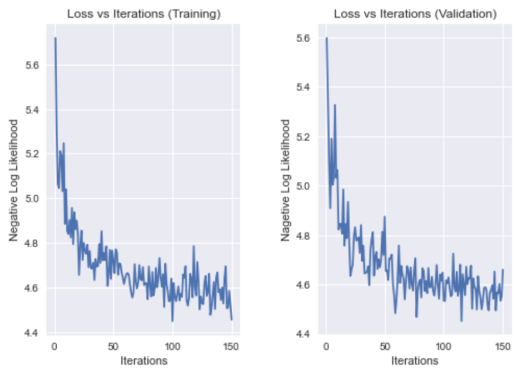

Figure 1: Attention model loss plot over number of iterations, where iterations are the number of times a full batch was passed through in the training.

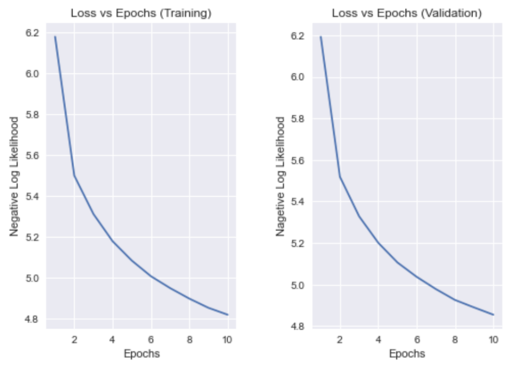

Figure 2: Transformer model loss plot over number of epochs.

The hyperparameters for the attention model were chosen based on the results of a grid search. The values considered were:

- batch size: 256, 512, 1024
- learning rates: 0.0001, 0.0003, 0.0007
- teacher forcing ratios: 0, 0.5
- dropout: 0, 0.1

The best hyperparameter combination being a batch size of 1024, learning rate of 0.0001, a teacher forcing ratio of 0.5, a dropout of 0.1 and a hidden size of 200 (embedding dimension). As for the transformer model, due to the GPU VRAM available, our hyperparameter options were very limited, so grid search was not necessary. The hyperparameters chosen for the transformer ended up being a batch size of 4, with 2 encoder and decoder layers, 2 heads in the multi-head attention, a learning rate of 0.00001, a dropout probability of 0.1 and a hidden size of 50 (embedding dimension which was lower, in order to fit in the VRAM). Figure 1 shows the loss over the number of iterations, which in this case was 150 iterations of batch size 1024. This is equivalent to training on 153600 training pairs, which is less than the full training set. This is due to the very slow training time of the attention model which made it infeasible, within the limited time, to train further. The transformer however, as can be seen in figure 2 was trained under 10 epochs of the full training set. This was possible due to the parallelism offered by the transformer model's architecture, which made it much more computationally efficient during training.

In figure 1, the attention model seems to plateau at a validation loss of around 4.6. Whereas in figure 2, the transformer reaches a validation loss of around 4.8, but looking at the trend of the curve, its clear that the model is not close to convergence. This means that the transformer model could still be further improved if given more training time. [COMMENT: Show loss comparisons. Show ROUGE results. Show output sentences from each model, for the same input sequence.]

6 Conclusions

7 Electronically-available resources

ACL provides this description and accompanying style files at

`https://2021.aclweb.org/files/acl-ijcnlp2021-templates.zip`

We strongly recommend the use of these style files, which have been appropriately tailored for the ACL-IJCNLP 2021 proceedings. You can also start from the Overleaf template linked from the conference website.

LaTeX-specific details: The templates include the LaTeX2e source (`acl2021.tex`), the LaTeX2e style file used to format it (`acl2021.sty`), an ACL bibliography style (`acl_natbib.bst`), an example bibliography (`acl2021.bib`), and the bibliography for the ACL Anthology (`anthology.bib`).

8 Length of Submission

The conference accepts submissions of long papers and short papers. Long papers may consist of up to eight (8) pages of content plus unlimited pages for references and ethics/broader impact statement. Short papers may consist of up to four (4) pages of content, plus unlimited pages for references and ethics/broader impact statement. Upon acceptance, final versions of both long and short papers will be given one additional content page to address the reviewers' comments.

For both long and short papers, all illustrations and tables that are part of the main text must be accommodated within these page limits, observing the formatting instructions given in the present document. **Papers that do not conform to the specified length and formatting requirements are subject to be rejected without review.**

The conference encourages the submission of additional material that is relevant to the reviewers but not an integral part of the paper. There are two such types of material: appendices, which can be read, and non-readable supplementary materials, often data or code. Additional material must be submitted as separate files, and must adhere to the same anonymity guidelines as the main paper. Appendices should be no longer than 4 pages. The paper must be self-contained: it is optional for reviewers to look at the supplementary material. The paper should not refer, for further detail, to documents, code or data resources that are not available to the reviewers. Refer to Section 12 for further information.

Workshop chairs may have different rules for allowed length and whether supplemental material is welcome. As always, the respective call for papers is the authoritative source.

9 Anonymity

As reviewing will be double-blind, papers submitted for review should not include any author information (such as names or affiliations). Furthermore, self-references that reveal the author's identity, *e.g.*,

> We previously showed (**?**) ...

should be avoided. Instead, use citations such as

> **?** previously showed...

Please do not use anonymous citations and do not include acknowledgements. **Papers that do not conform to these requirements may be rejected without review.**

Any preliminary non-archival versions of submitted papers should be listed in the submission form but not in the review version of the paper. Reviewers are generally aware that authors may present preliminary versions of their work in other venues, but will not be provided the list of previous presentations from the submission form.

Please do not include github links that reveal the authors' identities in your submission. If you feel it is important to include your source code, you can zip the code and submit it to softconf.

Once a paper has been accepted to the conference, the camera-ready version of the paper should include the author's names and affiliations, and is allowed to use self-references and provide the related github link.

LaTeX-specific details: For an anonymized submission, ensure that `\aclfinalcopy` at the top of this document is commented out, and that you have filled in the paper ID number (assigned during the submission process on softconf) where `***` appears in the `\def\aclpaperid{***}` definition at the top of this document. For a camera-ready submission, ensure that `\aclfinalcopy` at the top of this document is not commented out.

10 Multiple Submission Policy

ACL-IJCNLP 2021 will not consider any paper that is under review in a journal or another conference at the time of submission, and submitted papers must not be submitted elsewhere during the ACL-IJCNLP 2021 review period. This policy covers all refereed and archival conferences and workshops (e.g., NAACL). The only exception is that a paper can be dual-submitted to both ACL-IJCNLP 2021 and an ACL-IJCNLP workshop.

In addition, we will not consider any paper that overlaps significantly in content or results with papers that will be (or have been) published elsewhere. Authors submitting more than one paper to ACL-IJCNLP 2021 must ensure that their submissions do not overlap significantly ($> 25\%$) with each other in content or results.

11 Formatting Instructions

Manuscripts must be in two-column format. Exceptions to the two-column format include the title, authors' names and complete addresses, which must be centered at the top of the first page, and any full-width figures or tables (see the guidelines in Section 11.5). **Type single-spaced.** Start all pages directly under the top margin. The manuscript should be printed single-sided and its length should not exceed the maximum page limit described in Section 8. Pages should be numbered in the version submitted for review, but **pages should not be numbered in the camera-ready version**.

LaTeX-specific details: The style files will generate page numbers when `\aclfinalcopy` is commented out, and remove them otherwise.

11.1 File Format

For the production of the electronic manuscript you must use Adobe's Portable Document Format (PDF). Please make sure that your PDF file includes all the necessary fonts (especially tree diagrams, symbols, and fonts with Asian characters). When you print or create the PDF file, there is usually an option in your printer setup to include none, all or just non-standard fonts. Please make sure that you select the option of including ALL the fonts. **Before sending it, test your PDF by printing it from a computer different from the one where it was created.** Moreover, some word processors may generate very large PDF files, where each page is rendered as an image. Such images may reproduce poorly. In this case, try alternative ways to obtain the PDF. One way on some systems is to install a driver for a postscript printer, send your document to the printer specifying "Output to a file", then convert the file to PDF.

It is of utmost importance to specify the **A4 format** (21 cm x 29.7 cm) when formatting the paper. Print-outs of the PDF file on A4 paper should be identical to the hardcopy version. If you cannot meet the above requirements about the production of your electronic submission, please contact the publication chairs as soon as possible.

LaTeX-specific details: PDF files are usually produced from LaTeX using the `pdflatex` command. If your version of LaTeX produces Postscript files, `ps2pdf` or `dvipdf` can convert these to PDF. To ensure A4 format in LaTeX, use the command `\special{papersize=210mm,297mm}` in the LaTeX preamble (below the `\usepackage` commands) and use `dvipdf` and/or `pdflatex`; or specify `-t a4` when working with `dvips`.

11.2 Layout

Format manuscripts two columns to a page, in the manner these instructions are formatted. The exact dimensions for a page on A4 paper are:

- Left and right margins: 2.5 cm
- Top margin: 2.5 cm
- Bottom margin: 2.5 cm
- Column width: 7.7 cm
- Column height: 24.7 cm
- Gap between columns: 0.6 cm

Type of Text	Font Size	Style
paper title	15 pt	bold
author names	12 pt	bold
author affiliation	12 pt	
the word "Abstract"	12 pt	bold
section titles	12 pt	bold
subsection titles	11 pt	bold
document text	11 pt	
captions	10 pt	
abstract text	10 pt	
bibliography	10 pt	
footnotes	9 pt	

Table 1: Font guide.

Papers should not be submitted on any other paper size. If you cannot meet the above requirements about the production of your electronic submission, please contact the publication chairs above as soon as possible.

11.3 Fonts

For reasons of uniformity, Adobe's **Times Roman** font should be used. If Times Roman is unavailable, you may use Times New Roman or **Computer Modern Roman**.

Table 1 specifies what font sizes and styles must be used for each type of text in the manuscript.

LaTeX-specific details: To use Times Roman in LaTeX2e, put the following in the preamble:

```
\usepackage{times}
\usepackage{latexsym}
```

11.4 Ruler

A printed ruler (line numbers in the left and right margins of the article) should be presented in the version submitted for review, so that reviewers may comment on particular lines in the paper without circumlocution. The presence or absence of the ruler should not change the appearance of any other content on the page. The camera ready copy should not contain a ruler.

Reviewers: note that the ruler measurements may not align well with lines in the paper – this turns out to be very difficult to do well when the paper contains many figures and equations, and, when done, looks ugly. In most cases one would expect that the approximate location will be adequate, although you can also use fractional references (*e.g.*, this line ends at mark 295.5).

LaTeX-specific details: The style files will generate the ruler when `\aclfinalcopy` is commented out, and remove it otherwise.

11.5 Title and Authors

Center the title, author's name(s) and affiliation(s) across both columns. Do not use footnotes for affiliations. Place the title centered at the top of the first page, in a 15-point bold font. Long titles should be typed on two lines without a blank line intervening. Put the title 2.5 cm from the top of the page, followed by a blank line, then the author's names(s), and the affiliation on the following line. Do not use only initials for given names (middle initials are allowed). Do not format surnames in all capitals (*e.g.*, use "Mitchell" not "MITCHELL"). Do not format title and section headings in all capitals except for proper names (such as "BLEU") that are conventionally in all capitals. The affiliation should contain the author's complete address, and if possible, an electronic mail address.

The title, author names and addresses should be completely identical to those entered to the electronical paper submission website in order to maintain the consistency of author information among all publications of the conference. If they are different, the publication chairs may resolve the difference without consulting with you; so it is in your own interest to double-check that the information is consistent.

Start the body of the first page 7.5 cm from the top of the page. **Even in the anonymous version of the paper, you should maintain space for names and addresses so that they will fit in the final (accepted) version.**

11.6 Abstract

Use two-column format when you begin the abstract. Type the abstract at the beginning of the first column. The width of the abstract text should be smaller than the width of the columns for the text in the body of the paper by 0.6 cm on each side. Center the word **Abstract** in a 12 point bold font above the body of the abstract. The abstract should be a concise summary of the general thesis and conclusions of the paper. It should be no longer than 200 words. The abstract text should be in 10 point font.

Command	Output	Command	Output
`{\"a}`	ä	`{\c c}`	ç
`{\^e}`	ê	`{\u g}`	ğ
``{\`i}``	ì	`{\l}`	ł
`{\.I}`	İ	`{\~n}`	ñ
`{\o}`	ø	`{\H o}`	ő
`{\'u}`	ú	`{\v r}`	ř
`{\aa}`	å	`{\ss}`	ß

Table 2: Example commands for accented characters, to be used in, *e.g.*, BibTeX names.

11.7 Text

Begin typing the main body of the text immediately after the abstract, observing the two-column format as shown in the present document.

Indent 0.4 cm when starting a new paragraph.

11.8 Sections

Format section and subsection headings in the style shown on the present document. Use numbered sections (Arabic numerals) to facilitate cross references. Number subsections with the section number and the subsection number separated by a dot, in Arabic numerals.

11.9 Footnotes

Put footnotes at the bottom of the page and use 9 point font. They may be numbered or referred to by asterisks or other symbols.[2] Footnotes should be separated from the text by a line.[3]

11.10 Graphics

Place figures, tables, and photographs in the paper near where they are first discussed, rather than at the end, if possible. Wide illustrations may run across both columns. Color is allowed, but adhere to Section 13's guidelines on accessibility.

Captions: Provide a caption for every illustration; number each one sequentially in the form: "Figure 1. Caption of the Figure." "Table 1. Caption of the Table." Type the captions of the figures and tables below the body, using 10 point text. Captions should be placed below illustrations. Captions that are one line are centered (see Table 1). Captions longer than one line are left-aligned (see Table 2).

[2]This is how a footnote should appear.

[3]Note the line separating the footnotes from the text.

LaTeX-specific details: The style files are compatible with the caption and subcaption packages; do not add optional arguments. **Do not override the default caption sizes.**

11.11 Hyperlinks

Within-document and external hyperlinks are indicated with Dark Blue text, Color Hex #000099.

11.12 Citations

Citations within the text appear in parentheses as (**?**) or, if the author's name appears in the text itself, as **?**. Append lowercase letters to the year in cases of ambiguities. Treat double authors as in (**?**), but write as in (**?**) when more than two authors are involved. Collapse multiple citations as in (**? ?**).

Refrain from using full citations as sentence constituents. Instead of

> "(**?**) showed that ..."

write

> "**?** showed that ..."

LaTeX-specific details: Table 3 shows the syntax supported by the style files. We encourage you to use the natbib styles. You can use the command `\citet` (cite in text) to get "author (year)" citations as in **?**. You can use the command `\citep` (cite in parentheses) to get "(author, year)" citations as in (**?**). You can use the command `\citealp` (alternative cite without parentheses) to get "author year" citations (which is useful for using citations within parentheses, as in **?**).

11.13 References

Gather the full set of references together under the heading **References**; place the section before any Appendices. Arrange the references alphabetically by first author, rather than by order of occurrence in the text.

Provide as complete a citation as possible, using a consistent format, such as the one for *Computational Linguistics* or the one in the *Publication Manual of the American Psychological Association* (**?**). Use full names for authors, not just initials.

Submissions should accurately reference prior and related work, including code and data. If a piece of prior work appeared in multiple venues, the version that appeared in a refereed, archival venue should be referenced. If multiple versions of a piece of prior work exist, the one used by the authors should be referenced. Authors should not rely on automated citation indices to provide accurate references for prior and related work.

The following text cites various types of articles so that the references section of the present document will include them.

- Example article in journal: (**?**).
- Example article in proceedings, with location: (Börschinger and Johnson, 2011).
- Example article in proceedings, without location: (**?**).
- Example arxiv paper: (**?**).

LaTeX-specific details: The LaTeX and BibTeX style files provided roughly follow the American Psychological Association format. If your own bib file is named `acl2021.bib`, then placing the following before any appendices in your LaTeX file will generate the references section for you:

```
\bibliographystyle{acl_natbib}
\bibliography{acl2021}
```

You can obtain the complete ACL Anthology as a BibTeX file from `https://aclweb.org/anthology/anthology.bib.gz`. To include both the anthology and your own bib file, use the following instead of the above.

```
\bibliographystyle{acl_natbib}
\bibliography{anthology,acl2021}
```

11.14 Digital Object Identifiers

As part of our work to make ACL materials more widely used and cited outside of our discipline, ACL has registered as a CrossRef member, as a registrant of Digital Object Identifiers (DOIs), the standard for registering permanent URNs for referencing scholarly materials.

All camera-ready references are required to contain the appropriate DOIs (or, as a second resort, the hyperlinked ACL Anthology Identifier) to all cited works. Appropriate records should be found for most materials in the current ACL Anthology at `http://aclanthology.info/`. As examples, we cite (Goodman et al., 2016) to show you how papers with a DOI will appear in the bibliography. We cite (Harper, 2014) to show how papers without a DOI but with an ACL Anthology Identifier will appear in the bibliography.

Output	natbib command	Old ACL-style command
(?)	`\citep`	`\cite`
?	`\citealp`	no equivalent
?	`\citet`	`\newcite`
(?)	`\citeyearpar`	`\shortcite`

Table 3: Citation commands supported by the style file. The style is based on the natbib package and supports all natbib citation commands. It also supports commands defined in previous ACL style files for compatibility.

LaTeX-specific details: Please ensure that you use BibTeX records that contain DOI or URLs for any of the ACL materials that you reference. If the BibTeX file contains DOI fields, the paper title in the references section will appear as a hyperlink to the DOI, using the hyperref LaTeX package.

12 Supplementary Materials

Supplementary material may report preprocessing decisions, model parameters, and other details necessary for the replication of the experiments reported in the paper. Seemingly small preprocessing decisions can sometimes make a large difference in performance, so it is crucial to record such decisions to precisely characterize state-of-the-art methods.

Nonetheless, supplementary material should be supplementary (rather than central) to the paper. **Submissions that misuse the supplementary material may be rejected without review.** Supplementary material may include explanations or details of proofs or derivations that do not fit into the paper, lists of features or feature templates, sample inputs and outputs for a system, hyperparameter values, pseudo-code or source code, and data.

The paper should not rely on the supplementary material: while the paper may refer to and cite the supplementary material and the supplementary material will be available to the reviewers, they will not be asked to review the supplementary material.

Supplementary material can be uploaded as up to three separate files: a single appendix file in PDF format, a single .tgz or .zip archive containing software, and/or a single .tgz or .zip archive containing data.

12.1 Appendices

Appendices are material that can be read, and include lemmas, formulas, proofs, and tables that are not critical to the reading and understanding of the paper. In the submitted version, appendices should be uploaded as a separate file. **Submissions which include appendices in the main paper will be rejected without review.** In the camera ready version, instead, appendices should directly follow the text and the references.

LaTeX-specific details: In your camera ready version use `\appendix` before any appendix section to switch the section numbering over to letters.

12.2 Software and Data

Submissions may include software and data used in the work and described in the paper. Any accompanying software and/or data should include licenses and documentation of research review as appropriate.

13 Accessibility

In an effort to accommodate people who are color-blind (as well as those printing to paper), grayscale readability is strongly encouraged. Color is not forbidden, but authors should ensure that tables and figures do not rely solely on color to convey critical distinctions. A simple criterion: All curves and points in your figures should be clearly distinguishable without color.

14 Translation of non-English Terms

It is also advised to supplement non-English characters and terms with appropriate transliterations and/or translations since not all readers understand all such characters and terms. Inline transliteration or translation can be represented in the order of:

original-form
transliteration
“translation”

15 LaTeX Compilation Issues

You may encounter the following error during compilation:

> `\pdfendlink` ended up in different nesting level than `\pdfstartlink`.

This happens when `pdflatex` is used and a citation splits across a page boundary. To fix this, the style file contains a patch consisting of two lines: (1) `\RequirePackage{etoolbox}` (line 455 in `acl2021.sty`), and (2) A long line below (line 456 in `acl2021.sty`).

If you still encounter compilation issues even with the patch enabled, disable the patch by commenting the two lines, and then disable the `hyperref` package by loading the style file with the `nohyperref` option:

`\usepackage[nohyperref]{acl2021}`

Then recompile, find the problematic citation, and rewrite the sentence containing the citation. (See, *e.g.*, `http://tug.org/errors.html`)

References

Benjamin Börschinger and Mark Johnson. 2011. A particle filter algorithm for Bayesian wordsegmentation. In *Proceedings of the Australasian Language Technology Association Workshop 2011*, pages 10–18, Canberra, Australia.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. Noise reduction and targeted exploration in imitation learning for Abstract Meaning Representation parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–11, Berlin, Germany. Association for Computational Linguistics.

Mary Harper. 2014. Learning from 26 languages: Program management and science in the babel program. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, page 1, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Mahnaz Koupaee and William Yang Wang. 2018. Wikihow: A large scale text summarization dataset.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 605–612.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. 2014. Recurrent models of visual attention. *arXiv preprint arXiv:1406.6247*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.