Natural Language Understanding, Generation and Machine Translation - Week 7 - Summarisation

Antonio León Villares

April 2023

Contents

Based on:

- *Get To The Point: Summarisation with Pointer-Generator Networks, by See et al.*
- *Text Summarisation with Pretrained ENcoders, by Yang Liu and Mierella Lapata*
- *Conditional Generation with a Question-Answering Blueprint, by Narayan, S., et al.*
- *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, by Raffel, C., et al.*

1 Introduction to Natural Language Generation

- **What is the Natural Langauge Generation task?**
  - converting **linguistic** (or **non-linguistic**) inputs into a **linguistic** output
  - up to now, we've mainly considered **linguistic** inputs only (i.e machine translation)
- **What sort of non-lingusitic inputs can be processed by NLG systems?**
  - databases → report on structure
  - news articles → help messages
  - log files → summaries of bugs
  - images → captions
- **Why is NLG useful?**
  - in essence, it **facilitates** information access
  - nowadays, we have generated a lot of textual data, but a lot of information is also available in non-textual format
  - a single person won't be able to **process** all this data, let alone understand it
  - moreover, text is easier to understand than numbers/graphs/tables
  - NLG is thus everywhere: search engines, question-answering machines, speech synthesisers

Stock data						
04/10/96	103	101.25	101.625	32444	-74	5485
04/09/96	104	101.5	101.625	41839	-33	5560
04/08/96	103.875	101.875	103.75	46096	-88	5594
04/05/96	Holiday					
04/04/96	104.875	103.5	104.375	18101	-6	5682

Microsoft avoided the downwards trend of the Dow Jones average today. Confined trading by all investors occurred today. After shooting to a high of $104.87, its highest price so far for the month of April, Microsoft stock eased to finish at an enormous $104.37. The Dow closed after trading at a weak 5682, down 6 points.

Figure 1: The summary is much easier to understand, particularly for non-experts.

Team Stat Comparison		
1st Downs	19	22
Total Yards	338	379
Passing	246	306
Rushing	92	73
Penalties	16-149	7-46
3rd Down Conversions	4-13	6-16
4th Down Conversions	0-0	0-1
Turnovers	2	0
Possession	27:40	32:20

The New England Patriots lost two linebackers and two coaches in the offseason. They still know how to win thanks in large part to two stars they didn't lose. Tom Brady threw for 306 years and two touchdowns and Richard Seymour helped make a a game-turning defensive play as the Patriots opened their quest for an unprecedented third straight Super Bowl victory by beating Oakland 30–20 on Thursday night.

Figure 2: The summary is much easier to understand, particularly for non-experts.

a crowd of people on a beach flying kites.
a man flying kite in the middle of a crowded beach.
lots of people enjoying their time on the beach.

Figure 3: Image captioning is rather subjective, which showcases the difficulty of evaluation for NLG tasks.

Most blacks say MLK's vision fulfilled, poll finds WASHINGTON (CNN) – More than two-thirds of African-Americans believe Martin Luther King Jr.'s vision for race relations has been fulfilled, a CNN poll found – a figure up sharply from a survey in early 2008.

The CNN-Opinion Research Corp. survey was released Monday, a federal holiday honoring the slain civil rights leader and a day before Barack Obama is to be sworn in as the first black U.S. president.

The poll found 69 percent of blacks said King's vision has been fulfilled in the more than 45 years since his 1963 'I have a dream' speech – roughly double the 34 percent who agreed with that assessment in a similar poll taken last March.

But whites remain less optimistic, the survey found. 'Whites don't feel the same way – a majority of them say that the country has not yet fulfilled King's vision,' CNN polling director Keating Holland said. However, the number of whites saying the dream has been fulfilled has also gone up since March, from 35 percent to 46 percent.

Highlights

- 69 percent of blacks polled say Martin Luther King Jr's vision realized.
- Slim majority of whites say King's vision not fulfilled.
- King gave his "I have a dream" speech in 1963.

Figure 4: The highlights are executive summaries produced by journalists, typically in a telegraphic manner which need not be grammatically correct (for example, there are verbs missing).

2 Summarisation with LSTMs

2.1 The Summarisation Task

- **How do conditional language models differ from standard language models?**
  - in **language models**, the next word is predicted given the previous ones:
    $$P(y_t \mid y_{1:t-1})$$
  - in **conditional language models**, this prediction is based both on the previous words and some additional input x:
    $$P(y_t \mid y_{1:t-1}, x)$$
  - for instance, in **machine translation**, x would be the **source sentence** to be translated
  - in **summarisation**, x will be the text to summarise
- **What is the concrete aim of summarisation?**
  - given an input text x, **generate** a **summary** y, which:
    - is **shorter**
    - contains the **key** points of x
- **What are the 2 types of summarisation?**
  1. **Single-Document**: write y from a **single** document x
  2. **Multi-Document**: write y from **multiple** documents $x_1, \ldots, x_n$. Typically, these should have **overlapping content**
- **When would one use multi-document summarisation?**
  - when a variety of sources need to be analysed to derive conclusions
  - for example, if we want to explain the connection between coffee and breast cancer, we would check a variety of studies
- **Which is easier, single or multi document summarisation?**
  - generally, **single-document summarisation** is easier, since there are less sources of information which need to be coalesced
  - one could argue that in **multi-document summarisation**, more data is available to determine what the **main points** are; however, this would require that the models **learn** about **repetition** and **paraphrasing**
  - moreover, with **multi-document summarisation**, training/decoding takes much longer, since a lot more tokens need to be considered
- **What are the 2 strategies which can be used for summarisation?**
  1. **Extractive Summarisation**: use parts of the **original** text to form a **summary** (typically by concatenating sentences)
  2. **Abstractive Summarisation**: use NLG to **generate** new text which summarises the original
- **How do extractive and abstractive summarisation compare?**
  - **extractive summarisation** is **easier** (copying sentences produces grammatically correct summaries); however, it is more **restrictive**, since it can only copy (no paraphrasing), and the summaries might not be **coherent**
  - **abstractive summarisation** is **harder** (need to understand key points properly to paraphrase), but is more **flexible** and **human-like**

2.2 Get To The Point

2.2.1 Dataset

- **What is the CNN/Daily Mail Dataset?**
  - the first dataset which allowed for **neural networks** to try **summarisation** (see https://github.com/abisee/cnn-dailymail)
  - contains pairs of **news articles** ($\approx$ 800 words) and **summaries** dervied from the **story highlights** ($\approx$ 56 words)
  - the pairs were generated using:
    - $100k$ stories from **CNN**
    - $200k$ stories from **Daily Mail**
  - the **highlights** were 3-4 sentences written by **journalists** (in **telegraphic** manner), and were typically **independent** of each other (little co-referencing between each sentence)

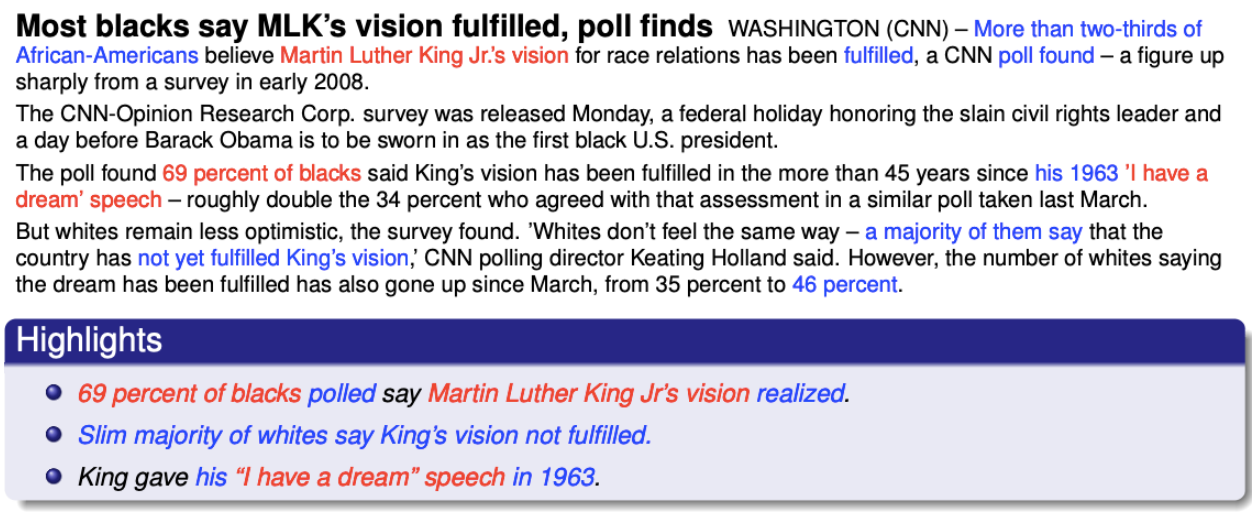


Figure 5: Parts of the highlights (red) are directly from the text, whereas other parts (blue) use **paraphrasing**.

2.2.2 Sequence-to-Sequence Attentional Model

- **What is the "Get To The Point" model?**
  - the first **successful** neural model for summarisation tasks
  - developed in 2017, was state-of-the-art until **transformers**
  - an **attentional** LSTM-based **encoder-decoder** model
- **What encoder does GTTP use?**
  - **single-layer**, **bidirectional** LSTM
  - the inputs were a **sequence of words** (instead of word-piece tokenisation)
  - **bidirectionality** helped the encoder not **forget** about the start/end of the inputs
  - generated a sequence of **hidden states**, $\underline{h}_1, \ldots, \underline{h}_n$
- **What decoder does GTTP use?**
  - **single-layer**, **left-to-right** LSTM

- as input, takes:
  - **previous** word embedding (during training, the word comes from the summary; during testing, the words is the previous word generated by the decoder)
  - a **decoder state** $\underline{s}_t$ (generated during decoding)
- needs **unidirectionality**, since we decoder from left to right

- **How did the attention mechanism work for GTTP?**
  - for the tth decoding step, attention was computed using:

$$e_i^t = \underline{v}^T \tanh(W_h \underline{h}_i + W_s s_t + \underline{b}_{attn})$$
$$\underline{a}^t = softmax(\underline{e}^t)$$

  - here:
    - $\underline{v}$
    - W_h
    - W_s
    - $\underline{b}_{attn}$

    are parameters learnt by the model
  - $\underline{h}_i$ denotes the **hidden state** of the ith input word

- **How did GTTP compute its vocabulary distribution?**
  - the **attention vector** was used alongside the **encoder hidden states** to generate a **context vector**:

$$\underline{h}_t^* = \sum_{i=1}^{n} a_i^t \underline{h}_i$$

  - this **context vector** gets **concatenated** with the **decoder state** $\underline{s}_t$, and gets fed through 2 linear layers, to generate a **vocabulary distribution**:

$$P_{vocab} = softmax(V_2(V_1 concat(\underline{s}_t, \underline{h}_t^*) + \underline{b}_1) + \underline{b}_2)$$

- **What training loss did GTTP use?**
  - **negative log-likelihood**
  - at each **decoding** step t, compute:

$$\ell_t = -\log P_{vocab}(w_t^*)$$

    where w_t^* denotes the tth word in the **reference summary**, and P_{vocab} computes the probability of said word under the model
  - the final loss is the average over all these losses:

$$\ell = \frac{1}{T} \sum_{t=0}^{T} \ell_t$$

- **What 2 flaws did the base sequence-to-sequence model have?**
  1. **Fixed Vocabulary**: if the **encoder** received an out of vocabulary word, the summarisation would be filled with `UNK` tokens
  2. **Repetition**: the attention mechanism was repetitive, which meant that certain sentences from the input were repeatedly copied/paraphrased by the model

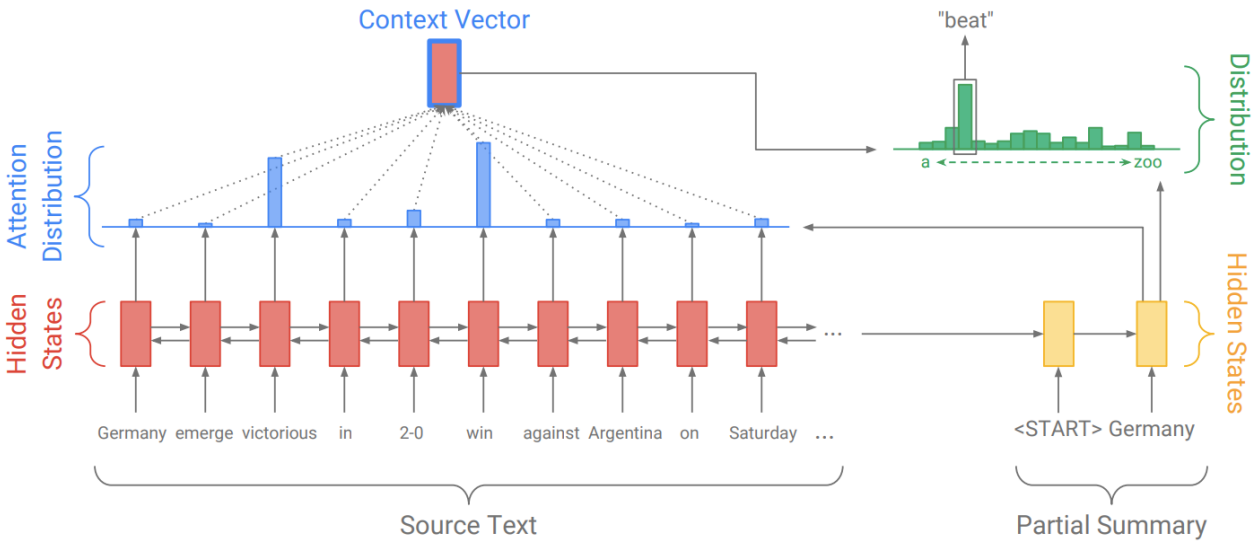


Figure 6: Illustration of the GTTP network.

2.2.3 The Pointer-Generator Network

- **What is the Pointer-Generator Network?**
  - deals with **unknown** words
  - for a given decoding step, decides whether:
    - we **copy** (point) a word directly from the input
    - we **generate** a word from the vocabulary
- **How does the Pointer-Generator Network determine when to copy or generate a word?**
  - at each decoder step, compute p_{gen}: the probability of **generating** the next word
  - p_{gen} is computed by using:
    - the **context** vector
    - the **decoder state**
    - the **decoder** input at the decoding step

    $$p_{gen} = \sigma(\underline{w}_{h^*}^T \underline{h}_t^* + \underline{w}_s^T s_t + \underline{w}_x^T \underline{x}_t + b_{ptr})$$

    (σ is the sigmoid function, and $\underline{w}_{h^*}, \underline{w}_s, \underline{w}_x, b_{ptr}$ are learnable parameters)
  - p_{gen} is used to define a **probability distribution** over an **extended vocabulary**: the union between the model's vocabulary, and all the word's appearing in the source document:

    $$P(w) = p_{gen} P_{vocab}(w) + (1 - p_{gen}) \sum_{i \,:\, w_i = w} a_i^t$$

    where w is a word from the extended vocabulary
  - notice that:
    - if w is **out of vocabulary**, P_{vocab} (the decoder probability) is 0
    - if w doesn't appear in the source document, then it won't have an attention weight, so $\sum_{i \,:\, w_i = w} a_i^t = 0$

2.2.4 The Coverage Mechanism

- **What is the purpose of the coverage mechanism?**
  - to **generate** less **repetitive** summaries
  - it penalises repeatedly **attending** to the same parts of the **source text**
- **What is the coverage vector?**
  - vector used to encompass which words have been attended to
  - at a given decoder step, we sum all the **previous** decoder attentions:

$$\underline{c}^t = \sum_{\tau=0}^{t-1} \underline{a}^\tau$$

  - this defines an **unnormalised** distribution over the attention weights used for each word in the source documents
  - the more attention it has received, the higher its coverage vector score
- **How is the coverage vector included in the attention mechanism to prevent repeated attention?**
  - to compute **attention**, we incorporate the **coverage vector**:

$$e_i^t = \underline{v}^T \tanh(W_h \underline{h}_i + W_s s_t + \underline{\boldsymbol{w}}_{\boldsymbol{c}} \boldsymbol{c}_{\boldsymbol{i}}^{\boldsymbol{t}} + \underline{b}_{attn})$$
$$\underline{a}^t = softmax(\underline{e}^t)$$

    where $\underline{w}_c$ is a vector of learnable parameters with the same shape as $\underline{v}$
  - this isn't sufficient, so the **coverage vector** is also used to define a **coverage loss**:

$$covloss_t = \sum_{i=1}^{n} \min(a_i^t, c_i^t)$$

    where:
    - if $a_i^t << c_i^t$, the ith word from the source has already been attended to a lot, so we'll want to decrease its attention weight (to minimise $covloss_t$)
    - if $a_i^t >> c_i^t$, then a given word hasn't been attended to much, and thus, can be incorporated into the summary
  - the final loss is a weighted sum of negative log likelihood and $covloss_t$:

$$\ell_t = -\log P(w_t^*) + \sum_{i=1}^{n} \min(a_i^t, c_i^t)\lambda$$

    where $P(w_t^*)$ is the Pointer-Generator Network distribution

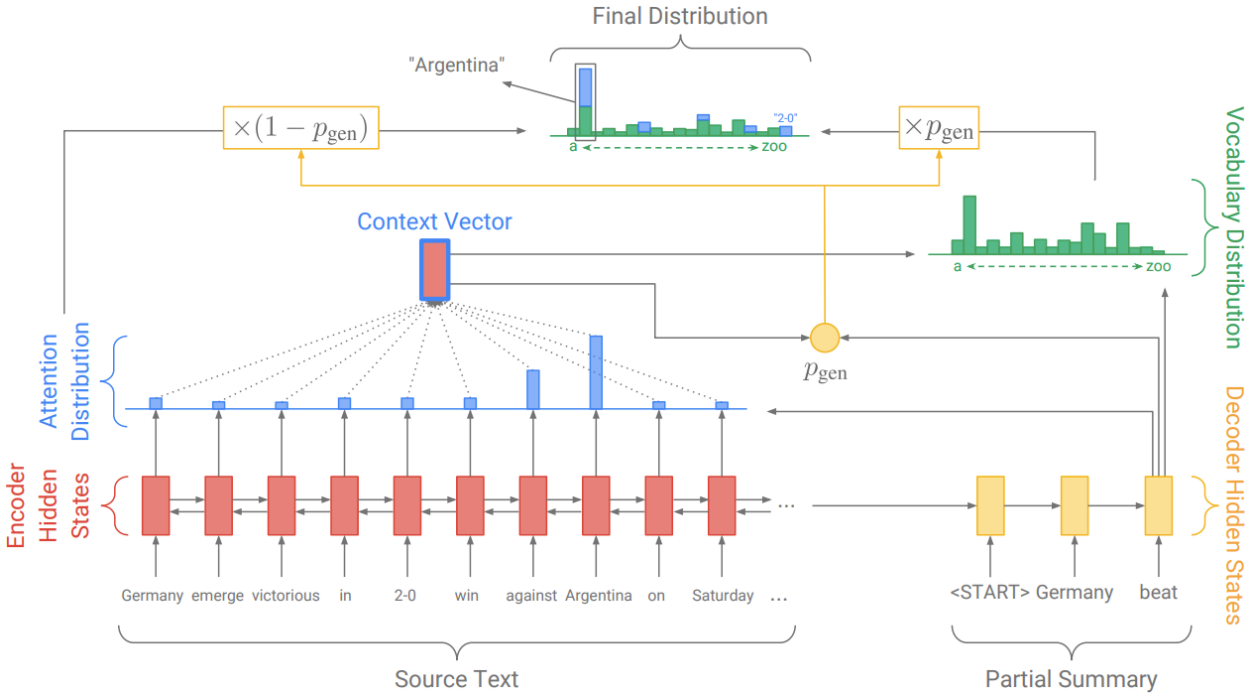


Figure 7: The final architecture for the GTTP network.

Original Text (truncated): lagos, nigeria (cnn) a day after winning nigeria's presidency, ***muhammadu buhari*** told cnn's christiane amanpour that **he plans to aggressively fight corruption that has long plagued nigeria** and go after the root of the nation's unrest. *buhari* said he'll "rapidly give attention" to curbing violence in the northeast part of nigeria, where the terrorist group boko haram operates. by cooperating with neighboring nations chad, cameroon and niger, **he said his administration is confident it will be able to thwart criminals** and others contributing to nigeria's instability. for the first time in nigeria's history, the opposition defeated the ruling party in democratic elections. *buhari* defeated incumbent goodluck jonathan by about 2 million votes, according to nigeria's independent national electoral commission. **the win comes after a long history of military rule, coups and botched attempts at democracy in africa's most populous nation.**

Baseline Seq2Seq + Attention: **UNK UNK** says his administration is confident it will be able to **destabilize nigeria's economy**. **UNK** says his administration is confident it will be able to thwart criminals and other **nigerians**. **he says the country has long nigeria and nigeria's economy.**

Pointer-Gen: *muhammadu buhari* says he plans to aggressively fight corruption **in the northeast part of nigeria**. he says he'll "rapidly give attention" to curbing violence **in the northeast part of nigeria**. he says his administration is confident it will be able to thwart criminals.

Pointer-Gen + Coverage: *muhammadu buhari* says he plans to aggressively fight corruption that has long plagued nigeria. he says his administration is confident it will be able to thwart criminals. the win comes after a long history of military rule, coups and botched attempts at democracy in africa's most populous nation.

Figure 8: Results for the different iterations of GTTP. In red, **factual errors** or **nonsensical sentences** from the baseline model. In green, **repetitions** by the pointer-generator model. In blue, parts of the **source** document which are copied into the summary.

- **How abstractive is the final model?**

- as can be seen above, as we add **improvements**, the model relies more and more on **copying**
- indeed, they found that the **baseline** produced more **novel** n-grams, but these were often erroneous

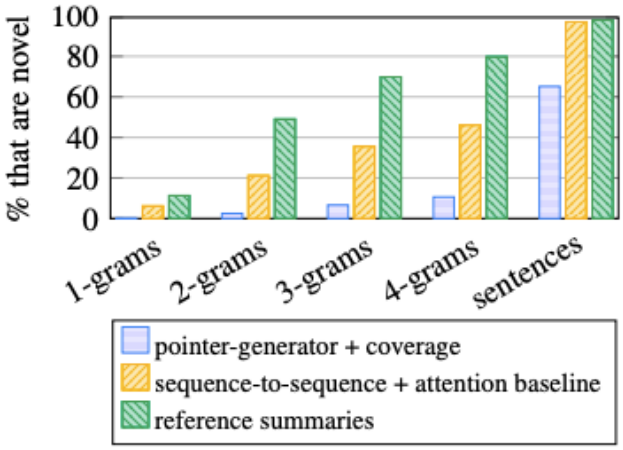


Figure 9: As model complexity increases, it tends to generate n-grams directly from source text.

Article: andy murray (...) is into the semi-finals of the miami open , but not before getting a scare from 21 year-old austrian dominic thiem, who pushed him to 4-4 in the second set before going down 3-6 6-4, 6-1 in an hour and three quarters. (...)
Summary: andy murray **defeated** dominic thiem 3-6 6-4, 6-1 in an hour and three quarters.

Article: (...) wayne rooney smashes home during manchester united 's 3-1 win over aston villa on saturday. (...)
Summary: manchester united **beat** aston villa 3-1 at old trafford on saturday.

Figure 10: Example summaries generated by the final model. In blue, novel words generated by the model.

3 Evaluating Summarisation: ROUGE

- **What is ROUGE?**
  - **automatic evaluation** method for **summarisation**
  - stands for **R**ecall-**O**riented **U**nderstudy for **G**isting **E**valuation
- **How is ROUGE computed?**
  - there are a variety of ROUGE metrics which can be computed
  - generally, they try to capture the number of **overlapping** n-grams between the **generated summary** and the **reference summary**
  - the **recall**-oriented ROUGE for a given n-gram is given by:

$$\texttt{ROUGE-n} = \frac{\text{number of matching n-grams between summary and reference}}{\text{number of n-grams in the reference}}$$

– if we have multiple references:

$$\texttt{ROUGE-n} = \frac{\sum_{S \in \text{Reference Summaries}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \text{Reference Summaries}} \sum_{gram_n \in S} Count(gram_n)}$$

– the **precision**-oriented ROUGE for a given n-gram is given by:

$$\texttt{ROUGE-n} = \frac{\text{number of matching n-grams between summary and reference}}{\text{number of n-grams in the summary}}$$

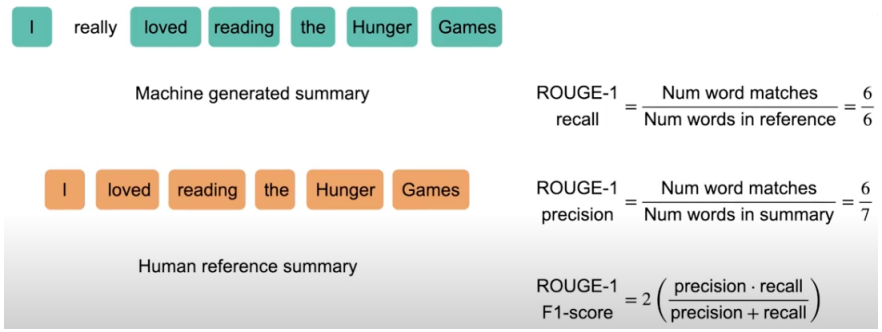
– we don't necessarily ahve to focus on n-grams, and can instead consider the **longest common subsequence**: the longest sequence of overlap between the summary and the reference:

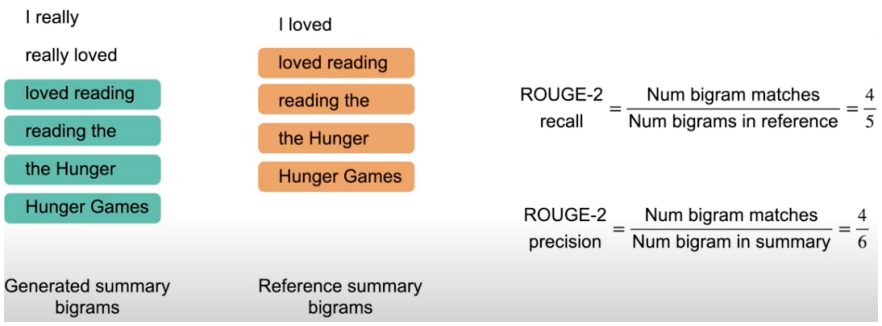
$$\texttt{ROUGE-L-precision} = \frac{\text{length of longest common subsequence between summary and reference}}{\text{number of words in the summary}}$$

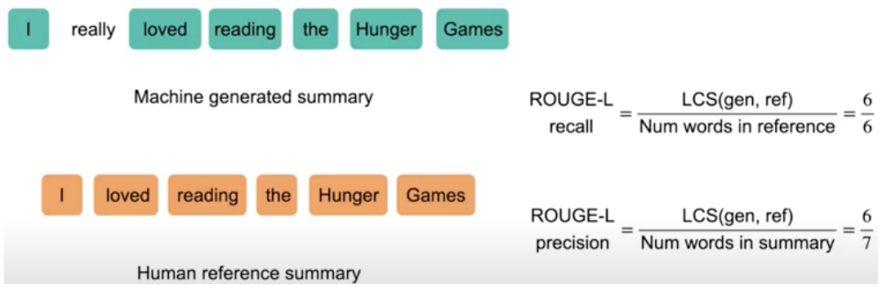
$$\texttt{ROUGE-L-recall} = \frac{\text{length of longest common subsequence between summary and reference}}{\text{number of words in the reference}}$$

– **recall** and **precision** ROUGE scores can be combined to obtain an F_1 score:

$$F_1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}}$$







- typically `ROUGE-1`, `ROUGE-2` and `ROUGE-L` are reported

- **Is `ROUGE` used a lot today?**
  - before, **summarisation models** were extremely poor at generating high-quality text, let alone summarisations
  - in these cases, `ROUGE` generally correlated well with human judgement (in terms of determinign if a summarisation was better than another one)
  - nowadays, the models are more adcvanced, so even if they generate gibberish (in terms of summarising), they might still obtain a high `ROUGE` score
  - much like with `BLEU`, `ROUGE` nowadays can't be trusted too much

4 Summarisation with Pretrained Transformers

4.1 `BERT`

4.1.1 `BERT` for Summarisation

- **Why can't `BERT` be used directly for summarisation?**
  - there are 2 main reasons behind why `BERT` can't be directly used for summarisation:
    1. **Information Representation**: the representations produced by `BERT` are created using **sentence-level** information (i.e it uses 2 sentences, masks them and then tries to predict their correct order). However, for **summarisation**, we require **document-level** awareness.
    2. **Architecture**: `BERT` generates good representations for tokens, but isn't designed to actually **generate** an output (it is just an **encoder**)

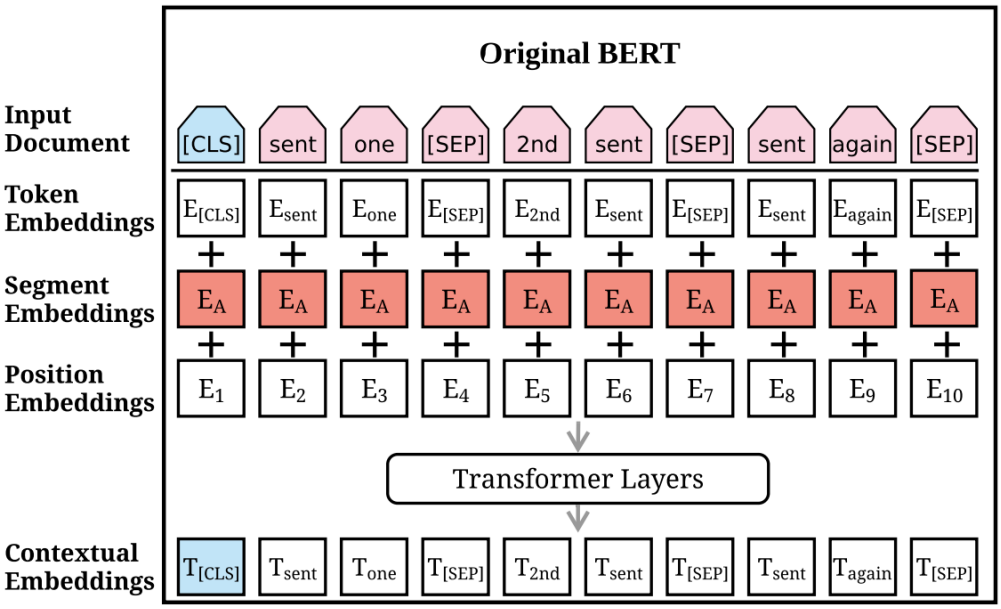
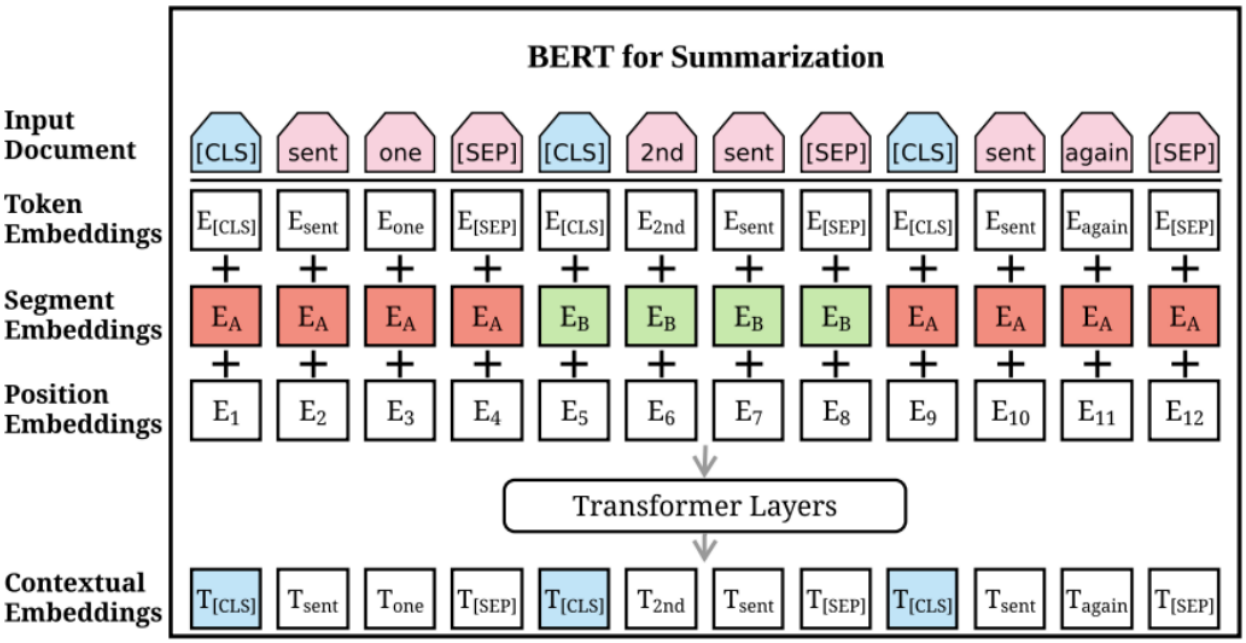


Figure 11: The original `BERT` encoder.

  - these were addressed in Text Summarization with Pretrained Encoders

- **How can we adapt `BERT` to encode document-level information?**
  - 2 main changes were included:

1. **Class Tokens for Each Sentence**: since documents are composed of several sentences, for the model to be aware of this, each sentence in the document is preceded by a `[CLS]` token. `BERT` learns to encode information for each sentence in the $T_{[CLS]}$.
2. **Segment Embeddings**: alternating **segement embeddings** are added to each word of a sentence. This allows the model to understand that the units composing the documents are the sentences (since each token in a sentence will have the same segment embedding added)



- **How can we adapt `BERT` for decoding?**
  - we need to add a **transformer decoder** at the end of the **encoder**
  - the **decoder** is trained from **scratch** during fine-tuning
- **Why shouldn't the encoder and decoder be fine-tuned in the same way?**
  - intuitively, the **encoder** has been pre-trained to be fairly good, so during fine-tuning, we shouldn't need much changes to the weights
  - however, the **decoder** is completely new, so it should be trained more aggresively
  - to do this, we use a **learning rate schedule**:

$$\eta = \xi \cdot \min(step^{-0.5}, step \cdot warmup^{-1.5})$$

where *step* is the current training step, and:

* for the **encoder**, we require a **smaller learning rate** and a **longer warming-up**, to get the encoder used to the fine-tuning tasks, and slowly adapt its weights. For summarisation, they used:

$$\xi = 2 \times 10^{-3} \qquad warmup = 2 \times 10^{4}$$

* for the **decoder**, we require a **larger learning rate** and a **shorter warming-up**, to get the decoder to more quickly adapt to the fine-tuning task. For summarisation, they used:

$$\xi = 1 \times 10^{-1} \qquad warmup = 1 \times 10^{4}$$

- **The above architecture is designed for abstractive summarisation. How can we adapt it for extractive summarisation?**
  - 2 simple strategies:
    1. **Text Spans**: make the decoder output **spans** of the input texty, to denote sentences (or parts of sentences) to use in the summary
    2. **Classification Task**: define a simple classifier, which determines whether a given input sentence should or should not be in the summary.

4.1.2 Evaluating BERT for Summarisation

- **What can be used as baselines when evaluting the summarisation power of BERT?**
  - we can use the 3 iterations of the GTTP model
  - since we are training on **news articles** from CNN/Daily Mail, we can use the first 3-4 sentecnes as baseline summaries
  - it is common for journalists to summarise the essence of articles in the first few sentences, so this is a "low-effort" baseline
- **How high does BERT score in ROUGE, compared to these baselines?**
  - BERT scores significantly higher than any of the 4 baselines
  - in some cases, the difference is in more than 2 points, which is quite significant
  - however, the baseline which takes the first sentences of the article performs better than all the 3 GTTP baselines

	ROUGE		
Models	1	2	L
seq-to-seq+attn	31.33	11.81	28.83
pointer-generator	36.44	15.66	33.42
pointer-generator + coverage	39.53	17.28	36.38
lead-3 baseline	40.34	17.70	36.57
BERTSUMABS	41.72	19.39	38.76

- **How significant are these results?**
  - notice, the quality of the simple baseline (using first 3 article sentences) is an artifact of the data, since it is typical for the first few sentences to be realtively high quality sentences
  - moreover, the data is fairly **extractive** in nature: one can get fairly good summaries by just copy-pasting
  - another issue is that these models are just that: models; thus, they can produce **factually inaccurate** summaries, despite sounding **fluent** and obtaining a high ROUGE score (for example, imagine instead of writing "Lee Harvey Oswald murdered Kennedy", it wrote "Kennedy murdered Lee Harvey Oswald")
  - the best way to evaluate **summarisation** is ultimately through human intervention

4.2 T5

4.2.1 Training T5

- **What is the philosophy of $T5$ as a language model?**

- **everything is text**
- T5 was pretrained and fine-tuned to be able to handle a variety of NLP tasks
- however, unlike with standard NLG, both the **input** and **output** must be **linguistic**

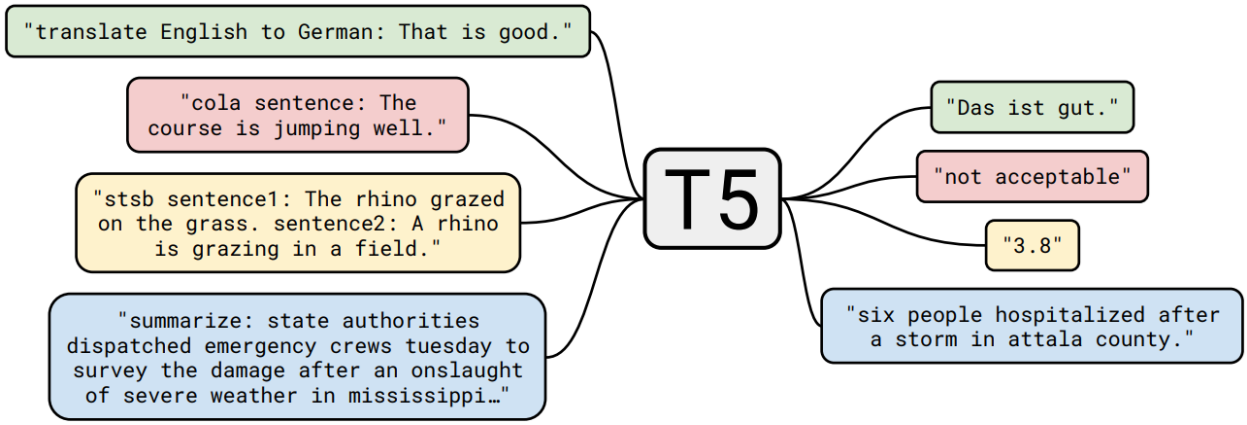


Figure 12: Recap of the capabilties of T5. Recall, the model was developed by Google in Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

- **What dataset was used to pretrain T5?**
  - the team at Google developed C4: A Colossal Cleaned Crawled Corpus
  - it consisted of hundreds of GB of **cleaned** data which had been crawled from the web
- **How exactly was T5 pretrained?**
  - inspired by BERT, T5 was pretrained in **masked token prediction**
  - T5 used **WordPiece** tokens
  - in particular, it converts 15% of the words in a sentence into **sentinel tokens**
  - any set of consecutively masked words get replaced by a **single sentinel token**
  - the model then needs to learn to predict what the masked tokens are

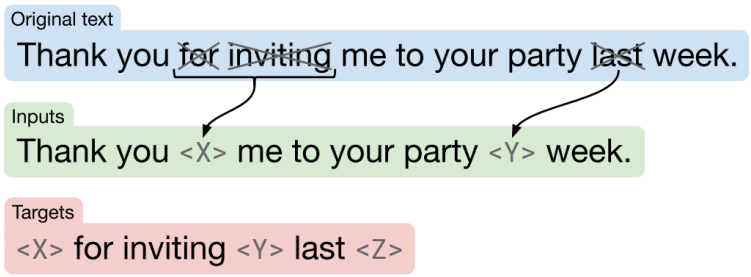
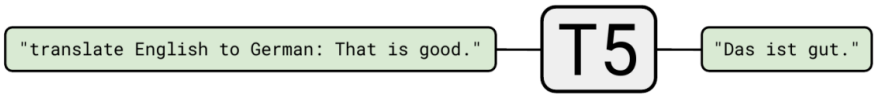


Figure 13: Here, "for", "inviting" and "last" were selected for masking. Since "for" and "inviting" are consecutive, they get replaced by a single sentinel token <X>. As an input during pretraining, the model obtains the masked input sentence. The target is an equivalent sentence, but with the masked words now visible, whereas the non-masked words are replaced with the sentinel token. An additional sentinel token <Z> is added to mark the end of the target sentence.

- **How was T5 fine-tuned?**

- a variety of common NLP tasks were used for fine tuning, including GLUE (which includes tasks like lingusitic acceptability, sentiment analysis, etc...), abstractive summarisation with CNN/Daily Mail, machine translation, etc ...
- the tasks were framed as **question-answering**, whereby the input was prefixed with a prompt
- for example, in translation from English to German:



4.2.2 Evaluating `T5` for Summarisation

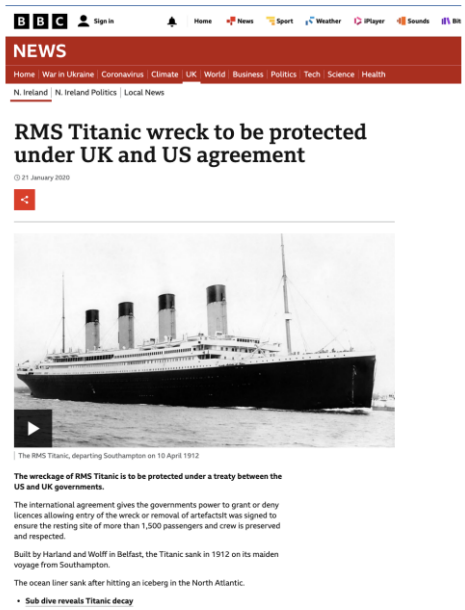
- **How well does `T5` perform in summarisation tasks?**
  - a variety of versions of `T5` were used for summarisation, using progressively larger, more resource-intensive models
  - whilst there was an improvement over `BERT` (for larger models, the difference could be greater than 2 `ROUGE`), arguably the difference wasn't that large
  - moreover, the difference between different `T5` models wasn't that impressive either, despite the exponential cost increase of handling such large models

	ROUGE		
Models	1	2	L
seq-to-seq+attn	31.33	11.81	28.83
pointer-generator	36.44	15.66	33.42
pointer-generator + coverage	39.53	17.28	36.38
lead-3 baseline	40.34	17.70	36.57
BERTSUMABS	41.72	19.39	38.76
T5-Small	41.12	19.56	38.35
T5-Base	42.05	20.34	39.40
T5-Large	42.50	20.68	39.75
T5-3B	43.52	21.55	40.69

5 Summarisation with Blueprints

5.1 Issues with Previous Conditional Generation Models

- **Why aren't the curent models not ideal at summarisation?**
  - whilst LSTM/Transformed-based models are **fluent**, they have 2 key problems:
    1. **Faithfulness**: there is no guarantee that the outputted **summary** is **consistent** with the **reference text**. In particular, if we have **multi-document summarisation**, the models can **hallucinate** information.
    2. **Output Control**: we have no way of controlling the summaries (i.e we can't define how long we want them to be, or what parts of the text to focus on)
  - overall, these systems should be able to **correctly** synthesise a lot of information together, and making up information does more **harm** than good (for example, if we want to explore how opinions on the influence of coffee on breast cancer is, we should expect that a model understands how these opinions have varied over time, and hosuldn't make up information)



The UK and US have signed a treaty to protect the Titanic wreck.

The Titanic is to be given full international protection after the US signed a treaty to protect the wreck.

The Titanic is to become the first UK country to sign a treaty to protect the wreck.

2020 systems were fully faithful 27% of the time (Maynez et al., 2020)

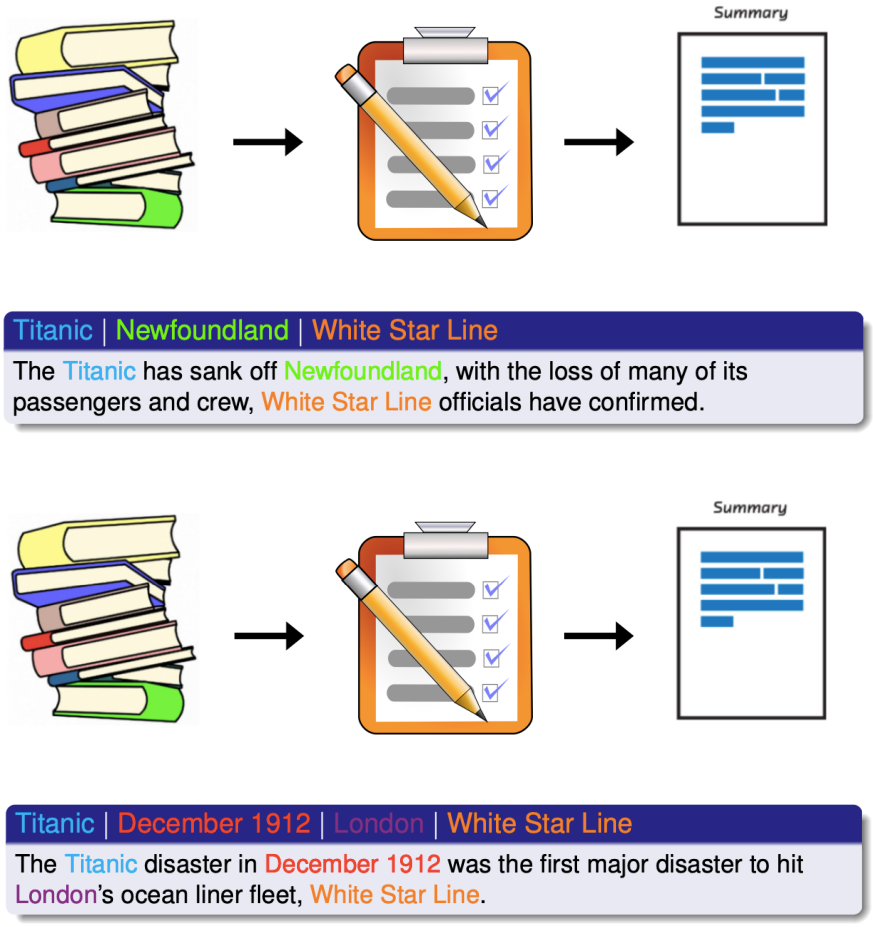
Figure 14: Systems from 2020 were bad at generating summaries consistent with the original text. In red, **factual mistakes** generated by some of the systems.

- **Why is it difficult to fix these issues?**
  - **neural networks** are **powerful**, but they are **black-box models**
  - we don't have access to ways of **tweaking** how these systems operate **directly**
- **What attempts have been made to fix the issues with conditional generation?**
  - Data-to-text Generation with Entity Modeling: change the way in which entities (i.e "Sheldon Cooper") are represented
  - Sticking to the Facts: Confident Decoding for Faithful Data-to-Text Generation: reduce **hallucinations** by ignoring tokens generated by the decoder with low confidence
  - Hierarchical Learning for Generation with Long Source Sequences: encode documents hierarchically (token, word, sentence, document levels)
  - Generating Long Sequences with Sparse Transformers: define **sparse self-attention**, to prevent the attention mechanism from attending to **all** the tokens in the input
  - Data-to-text Generation with Macro Planning and Planning with Learned Entity Prompts for Abstractive Summarization: introduce planning compoennts, which provide structure to the generated summaries. This is the approach we explore here.

5.2 Generating Question-Answering Blueprints

- **What is the core idea of planning?**
  - we provide the **summarisation** system with a **template**, which it can fill in to generate the summaries
- **How does planning work by using entity chains?**
  - we can identify **entities** within the summary task

- we can then ask the summarisation model to fill in the gaps (this was suggested in Planning with Learned Entity Prompts for Abstractive Summarization)



  - however, this has a problem: **entities** by themselves don't embody **content** (i.e "Titanic" can refer to a boat or a movie)
  - if we don't see the document, and only see the entities, we won't know what the summary will be about
  - without **context**, entities lack **specificity**

- **What alternative is there to planning, to ensure that content information is better reflected?**
  - we can frame the **planning** process as a **question-answering** task
  - this is motivated by the **Questions Under Discussion** (QUD) theory of **discourse structure**
  - we have a **partially structured** set of question, which **discourse participants** are **mutually committed** to resolving
  - **implicit questions** in the discourse get converted into what we speak about
  - for example, if someone comes into a lecture theatre, a **lecturer** will anticipate typical questions that a newcomer might have, and answer them preemptively (name, course structure, etc...)

- **How can we use QUD to generate plans?**

– we convert the **implicit** questions into **explicit**, and use these to define the **plan** (which we call a **blueprint**)
– for example, if we ask "What is the Titanic known for?", our **blueprint** will be a set of questions which will help **structure** the summary

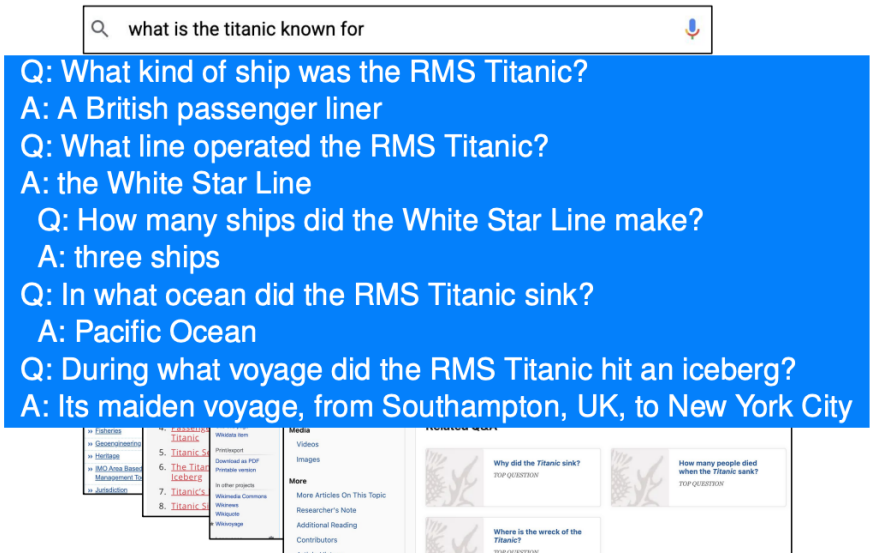


Figure 15: Example blueprint for the question "What is the Titanic known for?". This blueprint constitutes our plan. The questions and answers can be generated from a variety of documents.

- **What is the point of having a blueprint?**
  – it allows better **human supervision**, with regards to what the model is learning
  – we can look at the blueprint, and correct any faulty answers
  – based on the corrected blueprint, a **summary** can be generated, which should be more **factually correct**, and adapted to our requirements

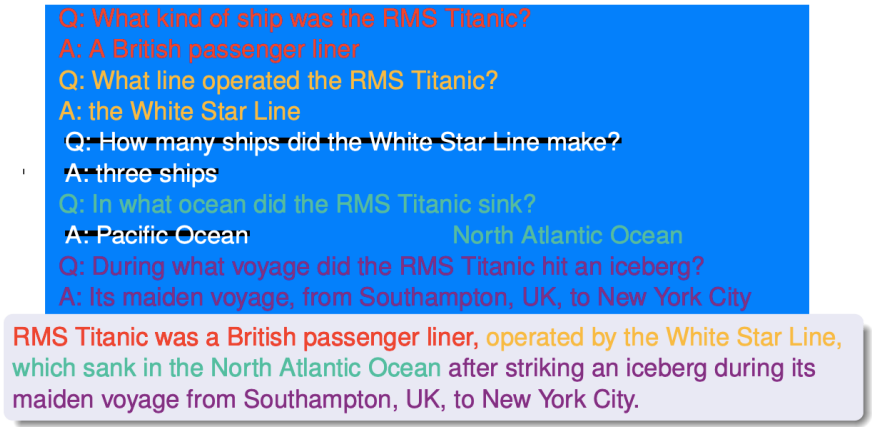


Figure 16: Certain Q/A pairs can be corrected (or even eliminated from the blueprint). Afterwards, a summary can be generated.

5.3 Blueprint Models

- **How can question-answer pairs be generated to create blueprints?**

– we follow a 6 step process
– it aims to generate pairs which are:
  * non-repeating
  * maximally different
  * capable of encompassing as much content information as possible
– for this:

1. **Generate Answers**: we can use trained models to identify **noun phrases** and **named entities**. These can be used as **candidate answers** for the questions.

The Shelby Mustang was built by Shelby American from 1965 to 1968, and from 1969 to 1970 by Ford.

2. **Generate Questions**: based on the **answers** above, we can **generate** corresponding question. To do this, we can use models trained on the `SQuAD` dataset, which focuses on question answering

Q: Who built the Shelby Mustang from 1965 to 1968?
A: Shelby American

Q: During what years was the Shelby Mustang built by Shelby American?
A: 1965 to 1968

Q: In what year did Ford take over production of the Shelby Mustang
A: 1969

3. **Duplicate Check**: these models tend to overgenerate questions, so we can reduce this number by firstly removing duplicate question-answer pairs.
4. **Round-Trip Consistency Check**: we can remove **questions** which don't produce **consistent** answers. For this, we can pose the question to the question-answering model; if it doesn't produce the expected answer (selected in (1)), we remove the question-answer pair.

5. **Rheme-Based Selection**: the **rheme** of a sentence is the part which provides new information. We can perform a **rheme check**, to ensure that we prioritise questions which seek out **new information**:

✓ Q: Who built the Shelby Mustang from 1965 to 1968?
A: Shelby American

✓ Q: During what years was the Shelby Mustang built by Shelby American?
A: 1965 to 1968

✗ Q: In what year did Ford take over production of the Shelby Mustang?
A: 1969

6. **Coverage**: the last filter prioritises **informative** question-answer pairs, by selecting those which are **non-overlapping** (and thus produce the widest coverage of information):

✓ Q: Who built the Shelby Mustang from 1965 to 1968?
A: Shelby American

✗ Q: During what years was the Shelby Mustang built by Shelby American?
A: 1965 to 1968

– overall, for a given text we thus obtain a set of high quality question-answer pairs which contain the most information about the text:

Q: Who built the Shelby Mustang from 1969 to 1970?
A: Ford
Q: During what years was the Shelby Mustang built by Shelby American?
A: 1965 to 1968
Q: In what year was the fifth generation of the Ford Mustang introduced?
A: 2005
Q: What was the Shelby Mustang revived as?
A: a new high-performance model

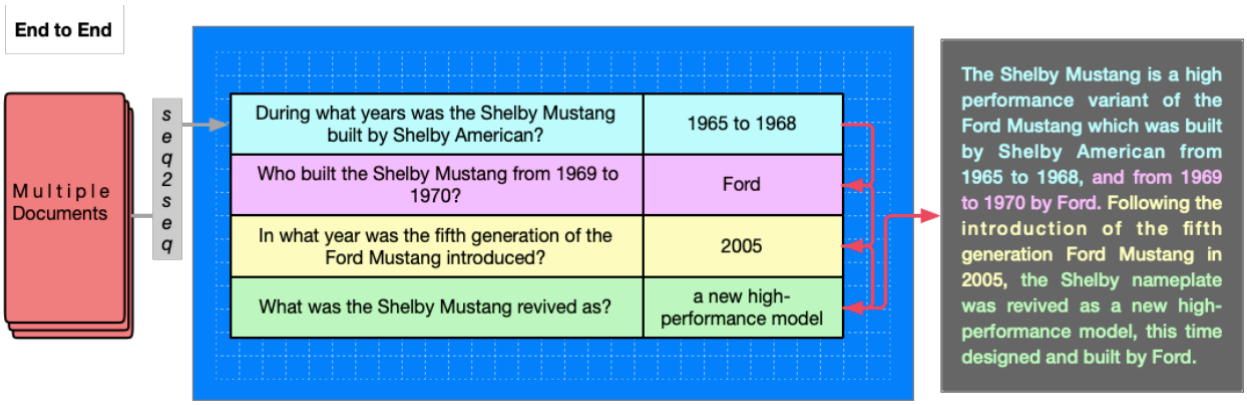
The Shelby Mustang is a high performance variant of the Ford Mustang which was built by Shelby American from 1965 to 1968, and from 1969 to 1970 by Ford. Following the introduction of the fifth generation Ford Mustang in 2005, the Shelby nameplate was revived as a new high-performance model, this time designed and built by Ford.

Over Generated Question-Answer Pairs		Round Trip	Rheme	Coverage
Q_1: What is a high performance variant of the Ford Mustang?	A_1: The Shelby Mustang	✓	✗	
Q_2: What is the high performance variant of the Ford Mustang called?	A_2: Shelby	✓	✗	
Q_3: What is a high performance variant of the Ford Mustang?	A_3: Shelby Mustang	✓	✗	
Q_4: What is a Shelby Mustang?	A_4: a high performance variant	✓	✗	
Q_5: The Shelby Mustang is a high performance variant of what?	A_5: the Ford Mustang	✓	✓	✗
Q_6: The Shelby Mustang is a high performance variant of what?	A_6: Ford Mustang	✓	✗	
Q_7: The Shelby Mustang is a high performance variant of what Ford model?	A_7: Mustang	✓	✗	
Q_8: Who built the Shelby Mustang from 1965 to 1968?	A_8: Shelby American	✓	✓	✗
Q_9: During what years was the Shelby Mustang built by Shelby American?	A_9: 1965 to 1968	✓	✓	✓
Q_{10}: In what year did Ford take over production of the Shelby Mustang?	A_{10}: 1969	✓	✗	
Q_{11}: What was the final year that Shelby American built the Mustang?	A_{11}: 1970	✗		
Q_{12}: Who built the Shelby Mustang from 1969 to 1970?	A_{12}: Ford	✓	✓	✓
Q_{13}: What event in 2005 led to the revival of the Shelby Mustang?	A_{13}: the introduction	✗		
Q_{14}: What generation of Mustang was introduced in 2005?	A_{14}: the fifth generation	✓	✗	
Q_{15}: What generation of Mustang was introduced in 2005?	A_{15}: fifth	✓	✗	
Q_{16}: In what year was the fifth generation of the Ford Mustang introduced?	A_{16}: 2005	✓	✓	✓
Q_{17}: What name was brought back for the 2005 Ford Mustang?	A_{17}: the Shelby nameplate	✓	✗	
Q_{18}: What was the Shelby Mustang revived as?	A_{18}: a new high-performance model	✓	✓	✓
$[$The Shelby Mustang is a high performance variant of the Ford Mustang$]_{P_1}$ which $[$was built by Shelby American$]_{P_2}$ $[$from 1965 to 1968,$]_{P_3}$ and $[$from 1969 to 1970 by Ford.$]_{P_4}$ $[$Following the introduction of the fifth generation Ford Mustang in 2005,$]_{P_5}$ $[$the Shelby nameplate was revived as a new high-performance model, this time designed and built by Ford.$]_{P_6}$				

5.3.1 End-to-End Blueprint Model

- **How does an End-to-End Blueprint Model incorporate blueprints into summarisation?**
  - it generates the **blueprint** and **summary** in one go
  - from the **input sequence**, a **blueprint** is generated

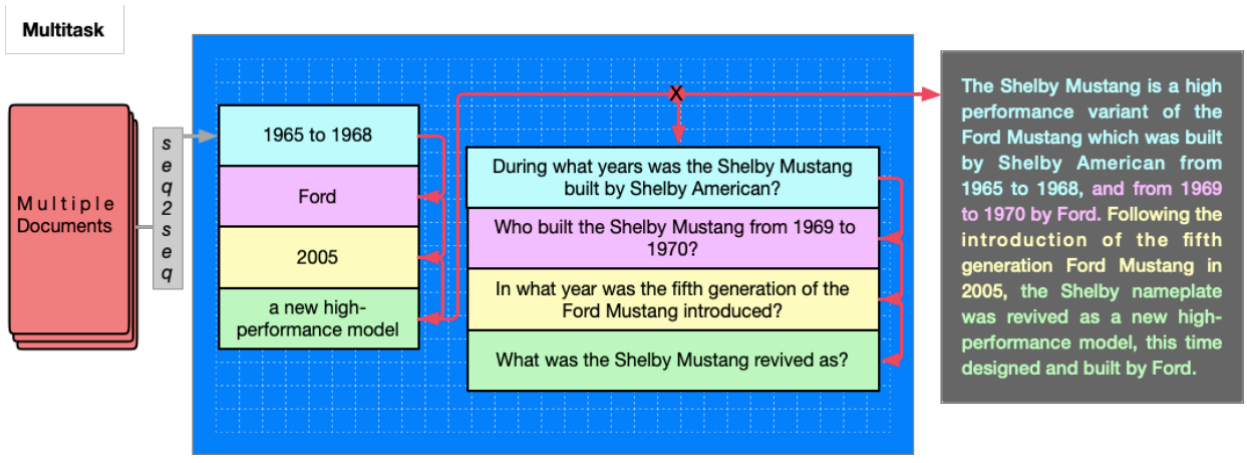
- then, from the **blueprint**, the **output summary** is produced (similarly to how in `T5` a **prompt** prefixes the input to specify the task at hand)
- both the **blueprint** and the **output summary** are returned



- **What are the flaws of an End-to-End Blueprint Model?**
  - the generated output is **too long** (blueprint + summary are outputted all at once)

5.3.2 Multitask Blueprint Model

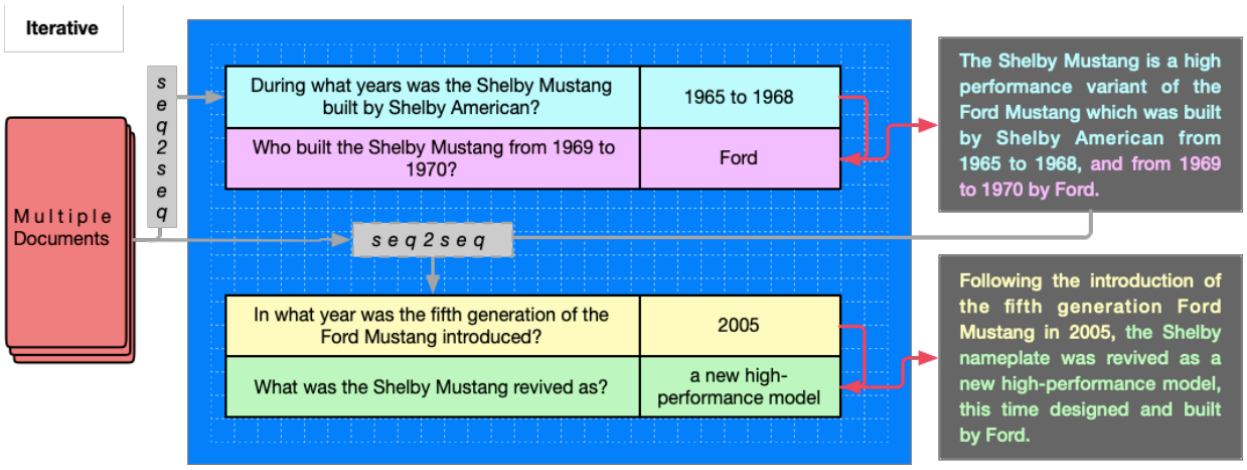
- **How does a Multitask Blueprint Model incorporate blueprints into summarisation?**
  - a **multitask blueprint model** can give 2 outputs:
    - the set of **answers**, alongside an **output summary**
    - the set of **answers**, alongside their corresponding **questions** (the **blueprint**)
  - this reduces the output sequence length, by generating the **output** and **blueprint** separately during **inference**



- **What are the flaws of a Multitask Blueprint Model?**
  - **blueprint** and **output** are no longer jointly trained; thus, the **output** solely relies on the **answers**, which might reduce generation quality
  - if we want the **blueprint**, we need to run the model twice (once for the summary output, once for the blueprint), but **blueprints** might vary between runs

5.3.3 Iterative Blueprint Model

- **How does an Iterative Blueprint Model incorporate blueprints into summarisation?**
  - the summary will no longer be generated in one go
  - each **output sentence** (and its corresponding **blueprint** question-answer pairs) are generated one at a time
  - the **output sentence** at time t depends on $t-1$ previous **output sentences**, alongside the **blueprint** at time t
  - since sentences are generated **incrementally**, this means we can potentially output summaries of **any** length which we desire
  - we also get that each sentence has its associated **blueprint**, so we can control performance through that aswell



- **What are the flaws of an Iterative Blueprint Model?**
  - we no longer have a **global plan**
  - it will be **slow**: the iterative process depends on the previous sentences, so we will spend more time decoding

5.4 Evaluating Summarisation with Blueprints

- **What datasets were used to evaluate the blueprint summarisation models?**
  - 3 datasets, used to test summarisation in different contexts:
    1. **AQusMuse** (AQUAMUSE: Automatically Generating Datasets for Query-Based Multi-Document Summarization): long-form question answering, simulates a a**search enginer**; the **answer** is based on multiple retrieved documents
    2. **WikiCatSum** (Generating Summaries with Topic Templates and Structured Convolutional Decoders): topic-focused **multi-document** summarisation, generats Wikipedia abstracts
    3. **SummScreen** (SummScreen: A Dataset for Abstractive Screenplay Summarization): **dialogue summarisation** for TV shows (i.e CSI, The Big Bang Theory)
  - the **blueprint** models were compared with a `LongT5` baseline (a `T5` model trained to handle longer token sequences - 4096 to be exact)

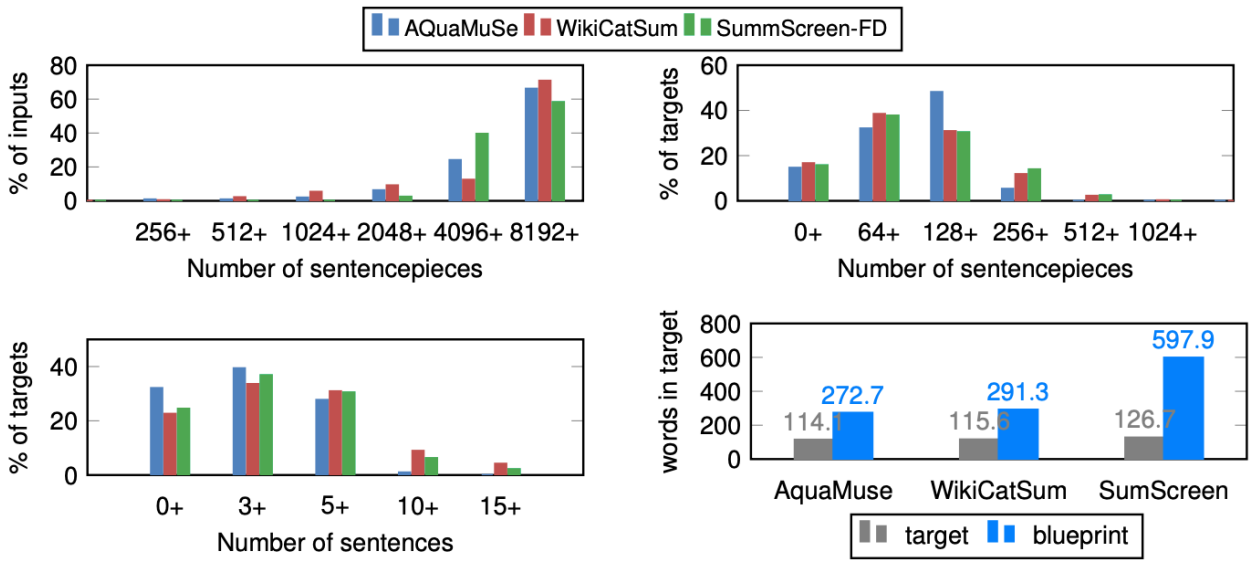
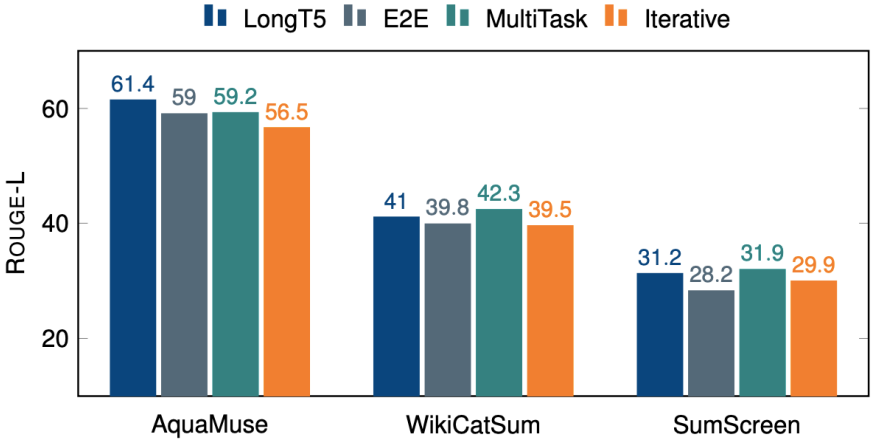


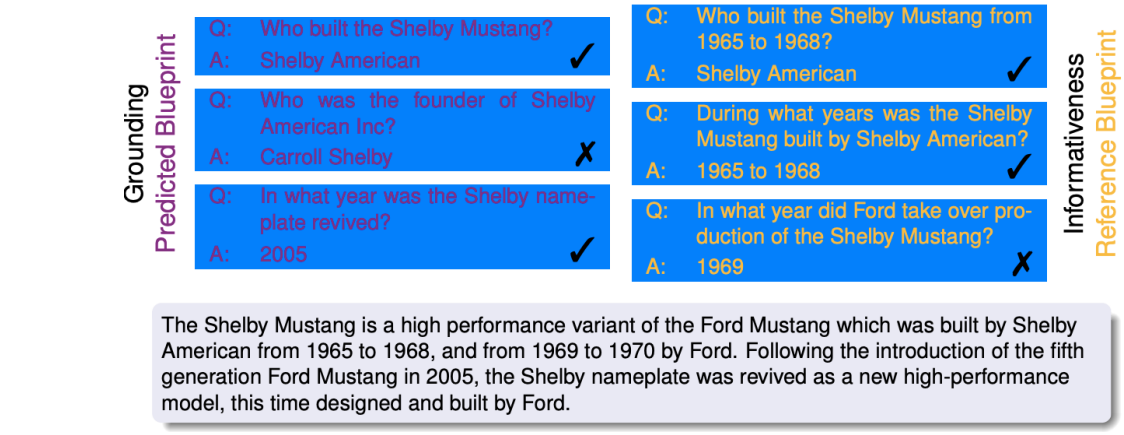
Figure 17: Distributions for the different datasets. In the top left, the number of tokens present within the input documents. In the top right, the number of tokens present in the reference summaries. In the bottom left, the number of sentences present in the reference summaries. In the bottom right, the number of words in both the reference summaries and the generated blueprints. Notice how muhc longer the blueprints are than the reference summaries.

- **How well do the models perform on `ROUGE`?**
  - whilst we know that `ROUGE` isn't the best evaluation metric (particularly with modern models), it can nonetheless be useful for evaluation
  - the 4 models were evaluated using `ROUGE-L`
  - except for `AquaMuse`, the **multitask** model obtained the best results (even above the `LongT5` model)
  - however, generally, all the models tended to obtain similar `ROUGE` scores



- **How can we use the question-answer structure of the blueprints to evaluate summarisation quality?**
  - we can evaluate how **grounded** the **summaries** are, by seeing whether the **output summary** can be used to correctly answer the **generated blueprints**

– if the **outputs** can do this **successfully**, this indicates that they are more **grounded**: they have been able to correctly distil information



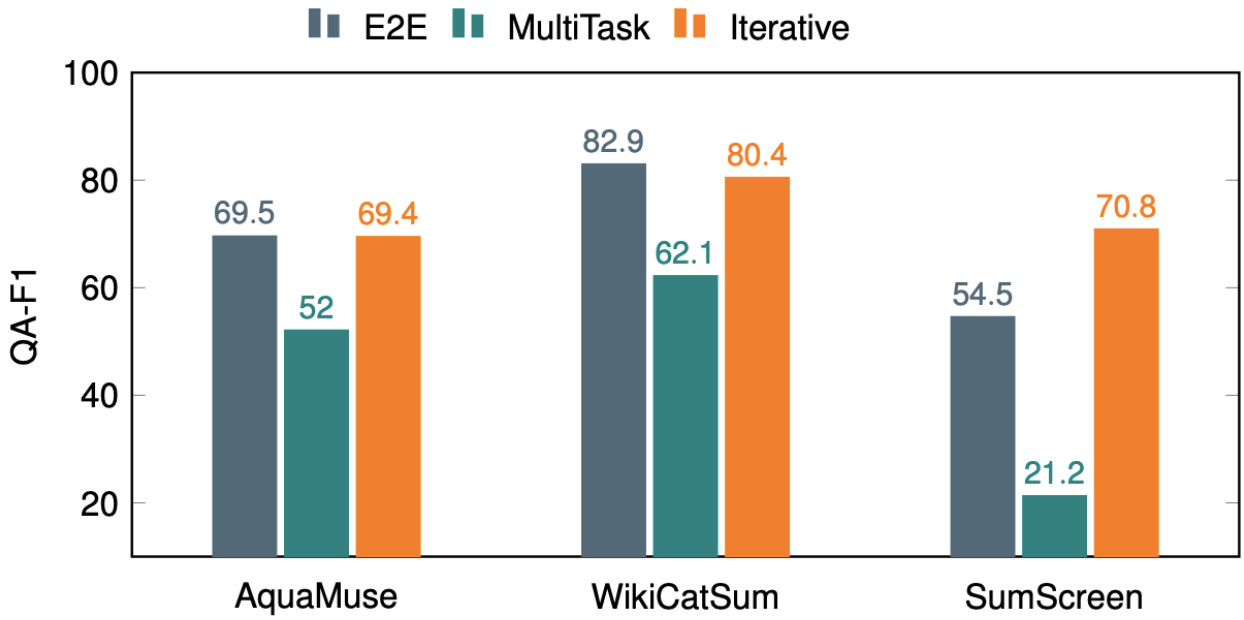


Figure 18: Results for the 3 **blueprint** models on each of the 3 datasets. Notice how the **multitask** model performs significantly worse. This is to be expected: summaries are generated by only looking at answers, since questions and answers are learnt separately in this model, so we shouldn't expect high groundedness.

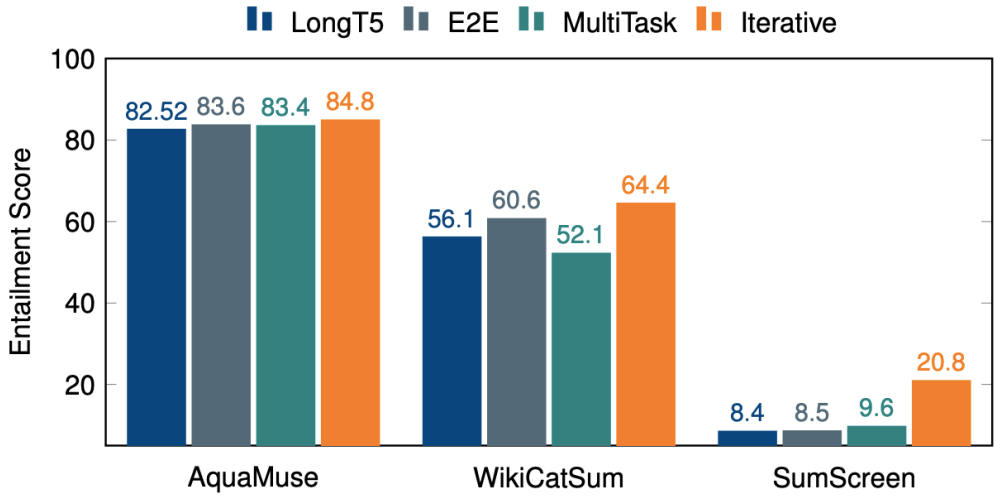
- **How can entailment be used to quantify the quality of the summaries?**
  - we can quantify whether the **output summaries** are **faithful** to the **input**, by using **textual entailment** (that is, can I infer a piece of text from some other piece of text?)
  - we can test:
    - whether the **blueprint** entails the **summary output**
    - whether the **input text** entails the **summary output**
  - a **higher entailment** indicates **higher faithfulness**

Text	Hypothesis	Entail
Regan attended a ceremony in Washington to commemorate the landings in Normandy.	Washington is located in Normandy.	False
Google files for its long awaited IPO.	Google goes public.	True
... a shootout at the Guadalajara airport in May, 1993, that killed Cardinal Juan Jesus Posadas Ocampo and six others.	Cardinal Juan Jesus Posadas Ocampo died in 1993.	True

- **How is entailment calculated?**
  - we can use **textual entaiment models**, which have been trained on public data:

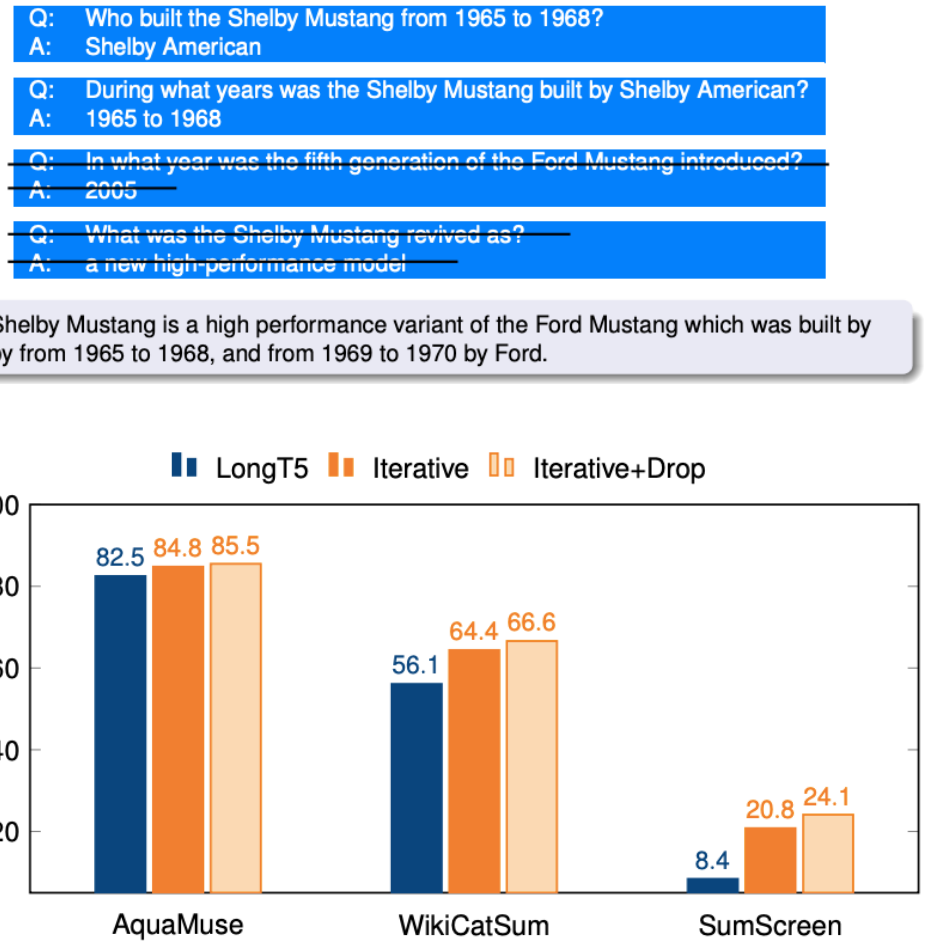
$$F(s) = \frac{1}{n} \sum_{i=1}^{n} E(D, s_i)$$

    where:
    - E is the **entailment model**
    - n is the number of sentences in the **summary**
    - D is the **input document(s)**
    - s_i is the ith sentence in the generated summary
  - empirically, $F(s)$ correlates well with human ratings
- **How faithful were the blueprint summaries found to be?**
  - generally, **blueprint** models obtained better scores than `LongT5`
  - in `SumScreen`, the **iterative** model performed significantly better (although performance was relatively poor througout)



- **How can we make the blueprint models more controllable?**
  - we still haven't been able to modulate the **length** of summaries produced
  - with the **iterative** model, this is possible

– we can choose to stop generating summary sentences, or select only some of the summary sentences



LongT5

Grissom and Catherine investigate when a man is found dead in a dumpster. They soon discover a lot more went on in the kitchen than cooking. Meanwhile Nick and Sara are called to the scene of a double homicide. The victims are a husband and his wife who were both in the process of selling off their rare records. Suspicion quickly falls on the wife's ex-boyfriend, but the evidence increasingly points to the husband.

Blueprint Iterative Model

Grissom, Catherine and David investigate when a man is found dead in a dumpster. The man had been eating at a restaurant called Aunt Jackpot's Pretzels. They discover that he ate himself to death. Meanwhile Nick and Sara look into the disappearance of a husband and wife who are found dead in their home. Also missing is a record collection that the husband had been collecting. They discover that the wife's neck was slashed in the attack. CSIs track down Missy Halter, a woman who helped them find the records.

Figure 19: A comparison between summaries between `LongT5` and the **iterative** model for a CSI: Las Vegas episode. Notice, `LongT5` includes **factually incorrect** information in its summary (in red). The **iterative** model doesn't do this; moreover, it provides useful details, in the form of named entities.