Sentence Fusion to Cohesive Text Generation

Philipp von Mengersen

A thesis presented for the Degree of
Bachelor of Computer Science

Faculty of Mathematics
and Computer Science,
Universität Leipzig

Referee: Jun.-Prof. Dr. Martin Potthast
Supervisor: Shahbaz Syed
Submission: 11.2021

Declaration

Unless otherwise indicated in the text or references, this thesis is entirely the product of my own scholarly work.

Leipzig, 2.11.2021

. . . . . .

Philipp von Mengersen

Acknowledgement

I would like to thank my primary supervisor, Shahbaz Syed, who guided me throughout this project and helped me with many decisions. I also want to thank friends and family, who did not tire of having the subject explained to them. And last but not least the Scientific Computing Center an University of Leipzig, which made it possible for me to run my software on their compute resources.

Abstract

This work examines a challenging extension of the conventional sentence fusion task of merging two sentences into a single sentence. Here, instead of only two sentences, we aim to fuse multiple related sentences into a cohesive text. We model this generation as an end-to-end learning task for sequence to sequence models. First, we compile a novel and a large scale dataset from two multi-document summarization datasets. Next, we finetune a state-of-the-art text summarization model on our dataset. Finally, we conduct extensive quantitative and qualitative evaluation following best practices for evaluating automatically generated text.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Nowadays most information is online. The web offers an unmanageable amount of documents for the individual user. Therefore tools are required that support users to navigate through the amount of documents, help them to seek the demanded information and prevent information overload. Tools for Automatic Summarization can assist users in a variety of ways and can be used in many situations: For instance in snippet generation for websites to provide a short overview of the content (Chen et al. 2020). It also can be used to get an overview about a scientific topic (Yasunaga et al. 2019) or to combine information from multiple news sources (Barzilay and McKeown 2005).

In this work we examine a mechanism that can support Automatic Summarization, it is called sentence fusion. When Humans write summaries, they tend to highlight important sentences and subsequently compress and connect them to new sentences in a summary (Lebanoff, K. Song, et al. 2019). The automatic mechanism of merging sentences was studied in previous works and first mentioned as sentence fusion by (Barzilay and McKeown 2005). Abstractive Summarization systems create new sentences with the relevant information of the input documents. These sentences do not appear in the original documents. The application of sentence fusion in abstractive summarization systems has beneficial impact on the generated summaries (Lebanoff, Muchovej, et al. 2020).

The task of sentence fusion is typically defined as combining few source sentences to a single target sentence or a short paragraph. In this work we are going to extend the fusion mechanism to apply it on multiple input sentences and create fused output texts with either a single or multiple output sentences. Such a fusion mechanism can increase the abstraction of the summary while preserving the gist of the content in a succinct manner and therefore improves the quality of summaries in terms of text cohesion and readability.

As with the original sentence fusion task, there are also additional fields of application, like retrieval based dialogue (Y. Song et al. 2018) or question answering (Li et al. 2018).

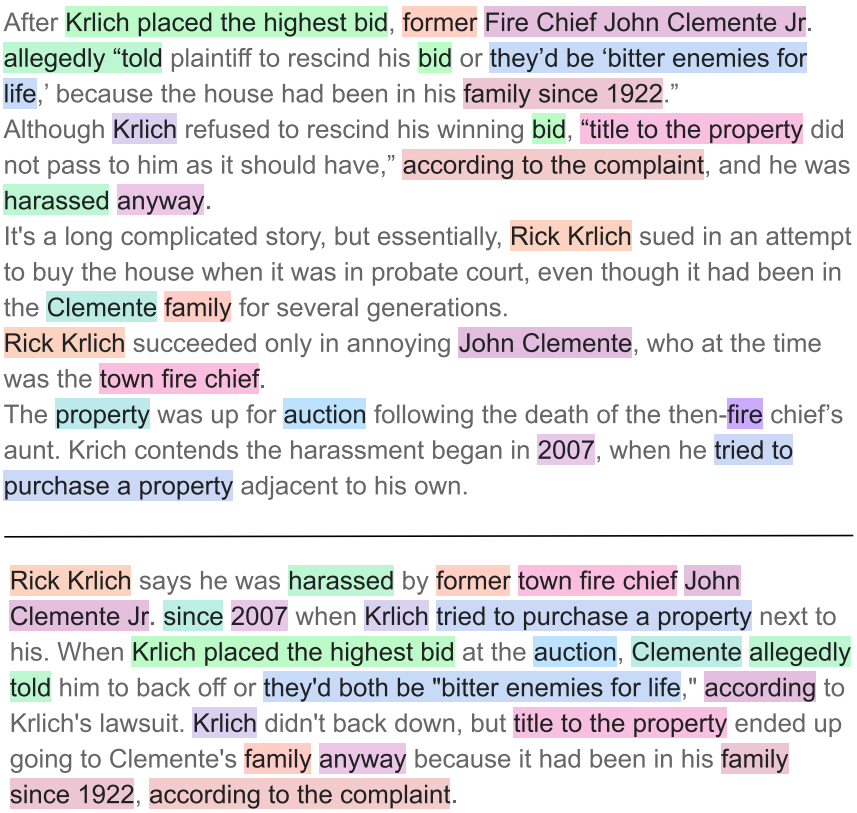

Figure 1.1: Example: extended sentence fusion
(top: input sentences; bottom: cohesive output text)

In this work we differentiate the extended sentence fusion task from related work (chapter: 2). We define several properties and present our approach to tackle the task (chapter: 3). We generate a large scale dataset (chapter: 4) that is sufficient to finetune a sequence-to-sequence neural network (chapter: 5). Finally we evaluate the training corpus and the performance of the model on the basis of several criteria (chapter: 6).

Chapter 2

Related Work

Unsupervised Approaches

The task of sentence fusion introduced by (Barzilay and McKeown 2005) is defined as connecting a bunch of sentences to a single sentence. Their presented algorithm outputs a single sentence that contains common information among the input sentences. Therefore it uses one of the input sentences as a centroid, an initial syntactic structure which gets aligned with the other sentences. Afterward the algorithm prunes superfluous parts of the syntactic structure and computes the final sentence. The resulting sentence could contain alternative wording from the other input sentences but is mainly based on the syntactic structure of the centroid. (Barzilay and McKeown 2005) apply sentence fusion in the multi-document summarization setting to provide the salient content in a single sentence.
(Marsi and Krahmer 2005) followed up on the proposed fusion algorithm and refined the alignment process to tackle further tasks of application, like question answering or information extraction.
The work of (Nayeem, T. A. Fuad, and Chali 2018) also picks up Barzialy et al.'s approach and extends it by adding paraphrases. The final sentence is selected by respecting information coverage as well as abstractiveness. The here proposed fusion algorithm is also applied in the multi-document summarization task.

Supervised Approaches

Next to unsupervised fusion algorithms also supervised approaches were described in previous works to tackle the sentence fusion task.
(T. Fuad et al. 2019) trained a the sequence-to-sequence Transformer model (Vaswani et al. 2017) on a machine generated dataset, constructed from the CNN/DailyMail

Corpus (Hermann et al. 2015). Each highlight sentence is mapped to similar sentences from the appropriate article, whereas the mapped source sentence are treated as the model input and the highlight sentence as the fusion output.
(Geva et al. 2019) also utilized the Transformer model, but their training data is generated in a different way. In their work they define several syntactic discourse connectives within text segments. They seek these segments in text collections and use a certain rule to resolve the connective and split the original segment into parts. With this technique they generate a large scale parallel corpus of sentences without discourse connectives and the connected counterparts. The described approach aims for inserting connectives into two syntactical unrelated sentences. The output consist of one or two sentences.
In (Lebanoff, K. Song, et al. 2019)'s approach for abstractive summarization also a neural based fusion mechanism is utilized to merge sentence pairs into a single sentence. They finetune a pretrained BERT (Devlin et al. 2019) on their sentence fusion downstream task. To generate examples of fused sentences for the training process, they exploit pairs of articles and summary. We use their described technique in our example generation pipeline as well. We go into this in more detail in chapter 3. Table 2.1 gives an overview of the previously developed sentence fusion models and their respective tasks.

model	**description**	**model output**	**data source**
supervised			
(Barzilay and McKeown 2005)	Merge common information from input sentences into output	single sentence	
(Marsi and Krahmer 2005)	Refined Version of Barzilay et al.'s Algorithm	single sentence	
(Nayeem, T. A. Fuad, and Chali 2018)	Additional Paraphrases	single sentence	
unsupervised			
(T. Fuad et al. 2019)	Supervised Training, Transformer	single sentence	CNN/ DailyMail
(Geva et al. 2019)	Insert Discourse Relations, Transformer	single sentence, two sentences	Wikipedia, Web articles about sport
(Lebanoff, K. Song, et al. 2019)	sentence fusion, BERT	single sentence	XSum, CNN/ DailyMail, DUC-04

Table 2.1: Overview of sentence fusion models

Differentiation from other tasks

We want to mention, that current abstractive summarization systems by themselves are also capable of producing output texts with several sentences. For instance, BART (Lewis et al. 2019) is a massively pretrained language model and can produce coherent and well structured text. If this model is finetuned on a summarization task with long output summaries, it also performs text transformations that result in a cohesive output text. However it combines this text transformation with content selection. This means, in abstractive summarization, the output summary is a compound of certain parts of the original document(s) and the choice of these parts is as important as the text transformation.
In this work we want to independently examine the text transformation that results in a cohesive output text, without the selection of relevant content and the target of compression.

Our extended sentence fusion task differentiates from all other tasks. Most previous works fuse a bunch of sentences to one or two sentences. In our case we have multiple sentences for input as well, but depending on the length of the input we expect an output text with one or more sentences that are connected through cohesive ties. In linguistics the term text refers "to any passage of whatever length, that does form a unified whole." Certain linguistic features allow a native speaker to easily decide whether a sequence of sentences is a text or not. Given following example from a cooking book: *Peel the potatoes. Afterward put them in hot water.* Here it is certain, that *them* refers to *potatoes*. This relation is called Anaphora and it ties both sentences together. All such linguistic features together help a sequence of sentences to be a cohesive text. (comp. Halliday and Hasan 1990)

Chapter 3

Task

This chapter defines several properties of the extended sentence fusion problem and presents our approach to tackle the task: We train a sequence-to-sequence model on a dataset, which consists of examples of the text transformation. In the following, we explain our procedure to generate training data and also show techniques in related work.

3.1 Definition

After performing sentence fusion on a sequence of sentences we expect that the resulting text covers the major information of the input sentences. Text transformations like reordering information, paraphrasing, insertion of new words or removal of superfluous words can be used to generate the cohesive output text. We define our fusion problem as follows:

- **Input**: Sentences that are not connected through cohesive ties and cannot be understand as a "whole". In our work we experiment with inputs varying between 2 and 11 sentences. The sentences can contain overlapping information.
- **Output**: A cohesive text with one or more sentences that can be understand as a "whole". The output text consists of the major information of the input sentences.
- **Input sentences**: Omission of non-critical information of the input in the output is tolerated. This means parts of the input sentences that only refer to the original context of the sentences and are not relevant in the new context of the compound of input sentences do not need to be conveyed in the output. This form of compression is to be minimal and should serve to create cohesion

rather than shortening the length of the input. Each individual input sentence is semantically related to the output text.

- **Output sentences**: The output sentences do not have to mimic the structure of the input sentences and can be abstract. The resulting text does not add new meaning compared with the input. As a result, each individual sentence of the output text is semantically related to the compound of input sentences.

We orientate us along the supervised approaches previously mentioned to accomplish our extended sentence fusion task.

3.2 Approach (Example Generation)

This section describes and justifies the general idea of the applied example generation procedure. As stated above is the task aiming for connecting multiple sentences to a cohesive text. The required training examples are pairs of input and output, also called source-target[1] pairs. Those fit the text transformation in that effect, that the source represents largely independent sentences and the target the cohesive text consisting of the relevant content of the source (comp. section 3.1).
The hypothesis is that such source-target pairs can be identified in a set of documents $(d_1, d_2, ..., d_n)$ about the same topic. Let T be the sentences from one document d_i (target-summary) and S all sentences from the remaining documents $(d_{(j_1)\neq i}, \ldots, d_{(j_{n-1})\neq i})$ (source articles) . Then a subset s $\subset$ S as source and a subset t $\subset$ T as target together can be a potential source-target pair. If the conditions are met, so if t is cohesive, the sentences in s are largely independent of each other and t conveys the relevant information of s then (s,t) is a source-target pair.
Since the target summary T is a closed text it is likely to find a target-text t $\subset$ T (t is also called anchor-text), that can be understood as a cohesive text segment. Furthermore are sentences of S and sentences of T about the same topic, hence it can be possible to find sentences in S that share information with t, in other words they have common parts that are semantically similar.
Our approach to find source-target pairs contains three steps. At first we identify anchor-texts in the target summary. Then we use the anchor-texts to retrieve source sentences in S. In the last step we seek a fitting combination among the queried source sentences. As a result, we expect source-target pairs that meet the criteria defined for our fusion problem (see section 3.1).

[1] In the following, we will sometimes refer to the input as the *source* and the output as the *target*

Example Generation Techniques in Related Work

(T. Fuad et al. 2019) and (Lebanoff, K. Song, et al. 2019) use similar approaches to generate their training examples. Both seek in pairs of articles and summary for fusion pairs that fit their task (see table 2.1). We refer to those pairs as (S,T) as described in the previous section. In the following, we name their approaches sentence mapping and sentence scoring.

Sentence mapping: (T. Fuad et al. 2019) map each sentence of T to all sentences of S with a Jaccard[2] score higher than the threshold value of 0.25. They determine this value by using a holdout set.

Sentence ranking: In (Lebanoff, K. Song, et al. 2019) a more elaborate technique is applied. They also map certain sentences of S to each sentence of T but employ a different selection procedure. They use the mean of ROUGE-1, -2, -L[3] as a similarity measure. Given an individual sentence t from T: (i) Find the most similar sentence s of S to t and remove s from S. (ii) Remove all overlapping content-words (tokens that are neither punctuation nor stop-words) between s and t from t. (iii) Repeat from step (i) until the remaining sentences in S have less than two common words with t.

The processing pipeline in this work is a combination of the two concepts of sentence mapping and sentence ranking.

[2]Jaccard Index: Let A, B be two sets, then their Jaccard index is $\frac{A \cap B}{A \cup B}$ (Wikipedia 2021b)

[3]ROUGE (Lin and Hovy 2002) is a frequently used package of metrics for automatic evaluation in text generation. ROUGE-1 respects uni-grams, ROUGE-2 respects bi-grams, ROUGE-L respects the longest common subsequence. In section 4.2.3.1 - metrics for containment, we describe how we use ROUGE for our purposes.

Chapter 4

Corpus Construction

In this chapter we introduce our sentence fusion corpus. It is created out of two text collections, namely Multi-News (Fabbri et al. 2019) and WCEP[1] (Ghalandari et al. 2020). Both are tackling the task of multi-document summarization. In the following we explain how we implemented the example generation procedure and give details about the created corpus.

4.1 Data Sources

Since (T. Fuad et al. 2019) and (Lebanoff, K. Song, et al. 2019) just have a single sentence as output instead of multiple target sentences, we need a source dataset with more coherent sentences in the target. Although the CNN/DailyMail corpus (Hermann et al. 2015) has multiple sentences in the summary it is not sufficient for us, because these sentences are just bullet points and can hardly be understand as a cohesive text. Hence we exploit two other text corpora, as already mentioned: Multi-News and WCEP. The following sections describe those corpora and the preprocessing before we fed instances into our pipeline.

4.1.1 MultiNews

Multi-News (Fabbri et al. 2019) is a large scale dataset for multi-document summarization on news articles. It includes 56 216 instances, each of them consists of multiple source articles and a human written summary. The source articles have together on average 82.73 sentences, the summary has on average 9.97 sentences and is a closed and cohesive text. Table 4.1 shows a record from the data.

[1]The dataset is from the Wikipedia Current Event Portal and is here referred to as WCEP

source 1 Meng Wanzhou, Huawei's chief financial officer and deputy chair, was arrested in Vancouver on 1 December. Details of the arrest have not been released...
source 2 A Chinese foreign ministry spokesman said on Thursday that Beijing had separately called on the US and Canada to "clarify the reasons for the detention"immediately and "immediately release the detained person". The spokesman...
source 3 Canadian officials have arrested Meng Wanzhou, the chief financial officer and deputy chair of the board for the Chinese tech giant Huawei,...Meng was arrested in Vancouver on Saturday and is being sought for extradition by the United States. A bail hearing has been set for Friday...
summary . . . Canadian authorities say she was being sought for extradition to the US , where the company is being investigated for possible violation of sanctions against Iran. Canada's justice department said Meng was arrested in Vancouver on Dec. 1... China's embassy in Ottawa released a statement.. "The Chinese side has lodged stern representations with the US and Canadian side, and urged them to immediately correct the wrongdoing"and restore Meng's freedom, the statement said...

Table 4.1: Multi-News example before preprocessing

4.1.2 Dataset from Wikipedia Current Events Portal (WCEP)

The second dataset WCEP (Ghalandari et al. 2020) contains 10200 article cluster, a set of articles that have the same topic as context. Each of them also includes a short ground truth summary, mostly a single or two sentence(s) text that summaries the general topic of the articles. We use the WCEP-100 version which limits each cluster to 100 articles, resulting in 650k articles in total.

4.1.3 Preprocessing

Multi-News: In our case short sentences are problematic, because they don't convey a lot of content and semantic similarity metrics wrongly score them as similar to other texts. Therefore we filter all sentences with less than 6 words in the preprocessing phase. We also avoid sentences that contain tokens like "Copyright", "AP", "Photo", "Image", "Caption". These tokens indicate sentences that are not connected to their context, provide metadata of the respective article or refer to images. We separate each text into its sentences and gather all sentences of source articles in one list. We also tokenize each sentence and assign Part-of-Speech tags which will be used later in the processing-pipeline. We finally pass the list of source sentences paired with the summary sentences to our example generation pipeline.

WCEP: In order to prevent redundant computation later in the pipeline, we eliminate duplicate sentences as early as possible. Therefore we remove all duplicate articles in each cluster. Comparing all articles in a cluster can be quite time

intensive. Hence we use the efficient LSH[2] algorithm (Leskovec, Rajaraman, and Ullman 2014) and map each article to its highly similar articles (approximated Jaccard threshold of 0.95). We then construct a graph of articles with edges indicating a duplicate relationship (Paullier 2020). We finally select one article of each connected component. This allows us to resolve duplicates that only differ in few characters. We further only use the unique articles and perform the following steps for each cluster: As we have done for the Mutli-News dataset we separate all articles into sentences, remove sentences with less than 6 words and critical tokens, and apply tokenization and POS-tagging. In order to get as many combinations of source sentences and target sentences, we pair each article with the remaining articles of the cluster: Let $(d_1, ..., d_n)$ be an article cluster with n documents. Then we generate all pairs $(d_1, (d_2, ..., d_n))$, $(d_2, (d_1, d_3, ..., d_n))$, ..., $(d_n, (d_1, ..., d_{n-1}))$. We handle the single article as the target and the remaining articles as source articles. We put all source sentences in one list and pass them together with the target sentences to the pipeline. Table 4.2 shows a preprocessed instance.

source sentences
• Meng Wanzhou, Huawei's chief financial officer and deputy chair, was arrested in Vancouver on 1 December.
• Details of the arrest have not been released...
• A Chinese foreign ministry spokesman said on Thursday that Beijing had separately called on the US and Canada to "clarify the reasons for the detention"immediately and "immediately release the detained person".
• Canadian officials have arrested Meng Wanzhou, the chief financial officer and deputy chair of the board for the Chinese tech giant Huawei,...
• ...Meng was arrested in Vancouver on Saturday and is being sought for extradition by the United States.
• A bail hearing has been set for Friday...
target sentences
• ... Canadian authorities say she was being sought for extradition to the US , where the company is being investigated for possible violation of sanctions against Iran.
• Canada's justice department said Meng was arrested in Vancouver on Dec. 1 ...
• China's embassy in Ottawa released a statement...
• "The Chinese side has lodged stern representations with the US and Canadian side, and urged them to immediately correct the wrongdoing"and restore Meng's freedom, the statement said...

Table 4.2: Multi-News example after preprocessing

[2]Local Sensitive Hashing allows to quickly find similar articles - here based on Jaccard Similarity - without comparing all documents

4.2 Example Generation Pipeline

The processing pipeline consists of three stages. We call these stages cohesive-text-detection, extract and select. As an abbreviation for cohesive-text-detection we just write detection. Each of these are successively passed.

The detection stage (see stage 1 in figure 4.1) takes an original target text T of a (S,T) pair as input and seeks for parts of T, that are valid target texts of a fusion pair. This stage encapsulates the logic for detection of cohesive text segments and returns the anchor-texts.

In the second processing step, the extraction stage (see stage 2 in figure 4.1) uses each anchor-text to query sentences of S that are semantically contained in the appropriate target text of T. The extraction stage guarantees that the extracted source sentences are related to their target text. It prevents unbearable Information loss in the final source-target pairs and regulates how sentences in the source are related to the target.

The final selection stage (see stage 3 in figure 4.1) selects for each valid target text a well suited set of source sentences among the previously extracted sentences of S. Although all extracted source sentences are related with the target text, it is still necessary to check whether the content in the target is covered by the source sentences. It could be the case, that the source sentences have many redundancies among them and only refer to one part of the target text.

4.2.1 Reasoning of Pipeline Architecture

The three pipeline stages can be ordered in multiple ways, because each stage independently deals with a certain task. The detection stage detects cohesive text segments, the extraction stage uses a text to query related sentences and the selection stage picks a well suited subset among the related sentences. This allows three orders which all come along with advantages and disadvantages. Following I argument why we decide on the above described architecture and point out the up- and downsides. Since the selection stage requires the sentences from the extraction stage, it has to be after the extraction stage. Beyond that, no further restrictions are given. In table 4.3 all possible orders of pipeline stages are listed with a description.To compare the three orders we examine the properties in which the orders differ.

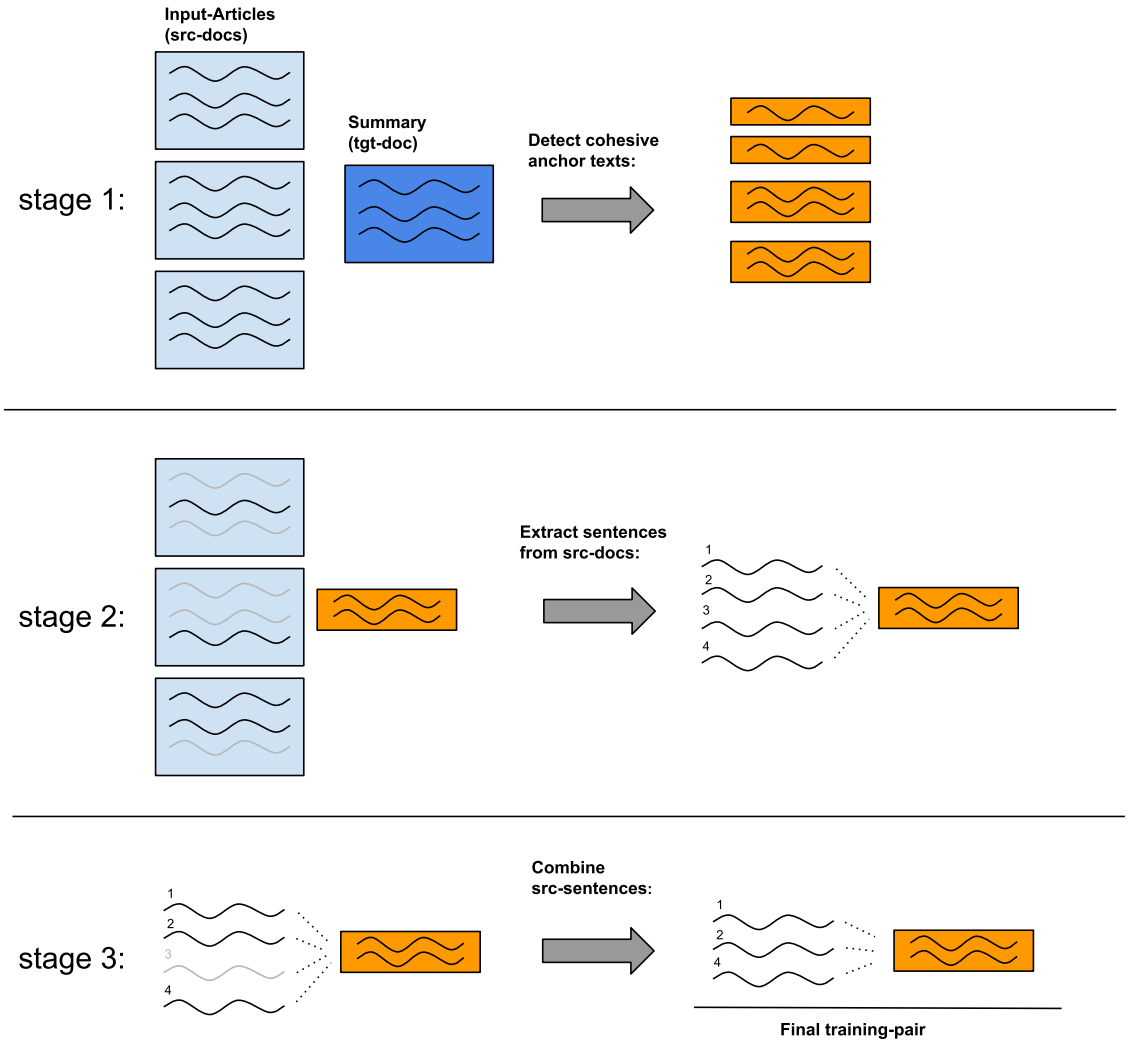

Figure 4.1: Pipeline Stages Overview

The first property is the computation of extracting source sentences. Order 1 detects cohesive target segments before they are used in the extraction stage. One target sentence can appear multiple times in a cohesive text segments and therefore can be part of multiple queries in the extraction. Thus source sentences could be queried with the same target sentence multiple times. Order 2 and order 3 fuse text segments after the extraction. Both use single target sentences to query source sentences. This prevents extracting source sentences with the same target sentence. This means that order 2 and 3 pass less redundant source sentences to the selection stage as order 1 and are therefore more efficient regarding computation.

Order	Description
(1) 1. Detect 2. Extract 3. Select	1. Identify cohesive text segments (anchor-texts) 2. Use the anchor as query to select single source-sentences 3. seek well-suited set of source sentences that covers the target (see section 3.1 - properties of fusion pairs)
(2) 1. Extract 2. Detect 3. Select	1. use individual sentences from the target text T as query to extract sentences from S 2. Detect cohesive text segments among the target sentences. If the target sentences get fused, unify their extracted source sentences (and remove duplicates). 3. seek a well-suited set of source sentences that covers the content of the target text (see section 3.1 - properties of fusion pairs)
(3) 1. Extract 2. Select 3. Detect	1. use individual sentences from the target text T as query to extract sentences from S 2. seek a well-suited set of source sentences that covers the content of the corresponding target sentence 3. Detect cohesive text segments among the target sentences. If target sentences get fused, unify their extracted source sentences (remove duplicates) and check if the content of the cohesive text segments is still covered (see section 3.1 - properties of fusion pairs)

Table 4.3: Pipeline architecture - possible orders

The second property is the query extent in extraction. This refers to the opportunity to extract relevant source sentences. Order 1 has a big query extent, because it can use the complete cohesive text segment to query source sentences. This offers the opportunity to extract sentences that are not related to individual target sentences, but to the complete cohesive text segment. How such relations come to pass is obvious to see, when we have a precise definition of semantic relations and a metric that recognizes them. For instance we could use a variation of Jaccard Index of words: the *word-containment*. This is quite similar to Jaccard Index, but the calculation differs in the numerator. We don't count the number of unique words in both texts, just the number of the unique target words. Let a, b be texts

and A, B the corresponding bag of words, then the word containment from A in B is: $word\text{-}containment(a, b) = \frac{|A \cap B|}{|B|}$. With this metric in use we can consider two text segments as related if the word-containment score is higher than a certain threshold. Table 4.4 shows an example.

- a: Anna is walking with her dog and telephones with her mother.
- b: This Saturday Anna goes for a walk, accompanied by her dog.
- c: During the walk she talks with her mother on phone.

⇒ a is related to the compound b-c, but not to the individual sentences c, b

Table 4.4: Sentence relation example

Order 2 and order 3 only have individual sentences in the query. Thus less sentences could be extracted as in order 1. If the extraction stage delivers more sentences it is more likely to find output-pairs in the selection stage. Furthermore are these relations with cross-sentence relation interesting training pairs. The model could also learn to distribute content of one sentence over multiple target sentences in the output.

The third property is the process of selecting source sentences. Order 1 and 2 detect cohesive target segments before the selection process. In their selection stage well-suited sets of extracted sentences are picked for the final training pair. In contrast order 3 selects extracted sentences for each individual target sentence. Subsequently these target sentences get fused to cohesive text segments. In this order an additional merge logic for already selected source sentences is necessary. In order 1 and 2 such a logic is not required.

Order / Property	extracting source sentences	query extent in extraction	process of selecting source sentences
1) detect - extract - select	inefficient	Big query extent	No additional logic
2) extract - detect - select	efficient	Small query extent	No additional logic
3) extract - select - detect	efficient	Small query extent	Additional logic

Table 4.5: Pipeline architecture - comparing orders

In table 4.5 all advantages and disadvantages are listed: The first order benefits from the big query extent and does not require the additional merge logic. Since the pipeline can run on sufficient compute resources the inefficient computation of extracting source sentences is bearable. Therefore we decide to build the pipeline using the first order detect - extract - select.

4.2.2 Cohesive text detection

The detection stage (see algorithm 1) outputs cohesive text segments that will be used as target in the final training pairs. As input it uses the complete target text T of a (S,T) input tuple and computes multiple cohesive text segments varying in the number of sentences. To identify cohesive text segments we utilize a straight forward approach. We set a range of numbers $m_1, m_2, \ldots, m_n$ as desired numbers of sentences in a cohesive segment. For each number m_i we seek all consecutive sentence pairs of length m_i. As a result we get paragraphs with m_i-sentences of a human written text. These paragraphs are cohesive texts and contain linguistic features that tie the sentences together. Since we just use parts of a text it could happen that the segment is not closed. This means the segment could refer to parts of the original text that are not within the segment. We ignore this effect because it rarely results in an insufficient outcome.

Algorithm 1 detect-cohesive-text

```
1:  Input
2:      m_1, ..., m_n                          ▷ number of desired sentences in segment
3:      tgt-text          ▷ list of sentences from the target text in original order
4:  Output
5:      segments                                        ▷ list of cohesive text segments
6:  procedure DETECT-COHESIVE-TEXT(m_1, ..., m_n, tgt-text)
7:      n-tgt-sents ← number of sentences in tgt-text
8:      segments ← empty list
9:      for m_i ← m_1, ..., m_n do
10:         upper ← n-tgt-sents - m_i
11:         for j ← 0, ... , upper do
12:             new-segment ← tgt-text[j, j + m_i]          ▷ indices: j, ..., j + m_i − 1
13:             push(segments, new-segment)
14:         end for
15:     end for
16:     return segments
17: end procedure
```

We set the number of target sentences $m_1, \ldots, m_n$ to 1, ..., 5. We don't use longer paragraphs because this would also result in bigger computation efforts in

the subsequent stages. Furthermore we think it is sufficient to test cohesive text generation limited to 5 output sentences to recognize performance and identify issues. Afterward experiments with longer target texts can be considered.
We discussed further approaches to identify cohesive paragraphs: (Lebanoff, K. Song, et al. 2019) use the BERT model (Devlin et al. 2019), which was, among other things[3], trained on the next sentence prediction task to score the likelihood that two sentences follow up on each other. We don't utilize this technique because the number of identified anchor texts is already high, as we observed in the stage 1 output. Furthermore it could be useful to exploit the annotation of a coreference resolution tool. Such a tool identifies connectives in and between sentences. However the functionality is not flawless and the lack in precision could add further errors (comp. Lebanoff, Muchovej, et al. 2020). Hence we decide to go without those approaches, which would require further testing.

4.2.3 Extraction

Following let t be a cohesive text segment from the detection stage and S all source sentences from the original input tuple (S,T). The Pipeline seeks in the extraction stage all source sentences from S that are semantically related with the segment t. Previously we already mentioned one way to determine semantic relations: the *word-containment* with a threshold (see section 4.2.1). In this chapter we describe our procedure for extraction and justify our chosen metric for semantic relations.

Code block 2 shows how the algorithm iterates over each source-sentence s of S and compares it with the segment. If the resulting score of the containment-metric is bigger than the threshold, s is considered as related. Additionally it is necessary to check if s appears in the segment. To avoid plane copying from source-sentences into the target we filter these sentences. The final model should rather learn new wording than exact copying. In order to do so, we check for each sentence in the anchor-segment, whether it is a duplicate of s or a subsequence (comp. algorithm 2, line 12). We go in more detail about this in the sections 4.2.3 *Duplicate detection* and 4.2.3 *Subsequence detection*. We already removed duplicate input articles in the preprocessing stage but many articles still contain the exact same sentences. We don't think that duplicate sentences in the input would fit any application use case of our sentences fusion mechanism. Therefore we also filter duplicates in related-sentences (comp. algorithm 2, line 13).

[3]BERT was also trained with masked documents to predict the missing words and learn the word context.

Algorithm 2 extract-sentences

```
1: Input
2:     metric                                  ▷ metric containment measure
3:     t                                                ▷ threshold value
4:     segment                          ▷ anchor text from extraction stage
5:     source-sents                     ▷ list of individual sentences from S
6: Output
7:     related-sents          ▷ list of sentences that are related to anchor
8: procedure EXTRACT-SENTS(metric, t, segment, source-sents)
9:     related-sents ← empty list
10:    for s ← source-sents do
11:        if metric(s,segment) ≥ t then
12:            if (not s ∈ segment) and
13:                (not s ∈ related-sents)  then
14:                push(related-sents, s)
15:            end if
16:        end if
17:    end for
18:    return related-sents
19: end procedure
```

Duplicate detection: To check if two sentences are equal we compute the edit distance[4] between them and compare the score against a threshold. We dynamically set the threshold value as follows: Let s_1, s_2 be two strings and l_1, l_2 their corresponding character length, then the dynamic threshold is $\frac{l_1+l_2}{20}$. If the score is below the threshold we consider the sentences as duplicates. This allows us to find strings that only differ in single characters like whitespace, punctuation or typos and through the dynamic threshold the tolerated number of equal character positions is depending on the length of the input strings.

Subsequence detection: Additionally we check if one string is almost a subsequence [5] of the other. Therefore we compute the length of longest-common-subsequence between them. Afterward we compare the portion of the subsequence in the sentences with the sentences themselves. We compare the bigger portion against a fixed threshold t: Let l_{lcs} be the length of the longest-common-subsequence between two strings s_1, s_2, and l_1, l_2 the corresponding character lengths, then we compare $\max\left(\frac{l_{lcs}}{l_1}, \frac{l_{lcs}}{l_2}\right) >= t$. If this expression is true we consider one of the sentences as a subsequence of the other. We set t to 0.9.

[4]The edit distance is a measure to quantify the difference between two strings. It is the minimal number edit-steps to transform one string into the other (comp. Navarro 2001).

[5]A subsequence of a string can be produced by deleting elements without changing the order of the elements (comp. Wikipedia 2021c)

We manually observed the performance of both metrics, duplicate-detection and subsequence-detection in various samples of data and get good results for both decider in our situation.

4.2.3.1 Containment-Metric

In the extraction procedure the containment-metric returns a score indicating to which degree the segment semantically contains the sentence. This task is similar to sentence similarity. The latter is defined as follows: Given two sentences as input then return a real number between zero and one that indicates the semantic similarity between both sentences, whereas the order of these sentences does not matter. In contrast, in sentence-containment the order of the argument impacts the outcome - table 4.6 gives an example for this.

• a: Anna is walking with her dog while telephoning with her mother.
• b: Anna talks with her mother on phone.
⇒ b is-contained-in a, but not vice versa

Table 4.6: Sentence containment example

Therefore methods for sentence similarity can not be adopted to sentence-containment without modification. The following part lists possible candidates for the containment metric. Afterwards we test each metric with various thresholds on a holdout-set to select the best performing candidate.

ROUGE-1-2-L: (Lebanoff, K. Song, et al. 2019) use the average of ROUGE-1, ROUGE-2 and ROUGE-L in their work. ROUGE (Lin and Hovy 2002) is a package of metrics used for automatic evaluation of generated text. It compares a reference text with a generated candidate text. ROUGE-1 compares all uni-grams (single words) of the reference and the candidate. The recall value of ROUGE-1 indicates which portion of uni-grams of the reference is covered by the candidate. It is the same for ROUGE-2 only that it does not refer to uni-grams instead it respects bi-grams. ROUGE-L computes the length of the longest sequence of common words (LCS) between reference and candidate. The recall of ROUGE-L indicates the portion of this LCS in the reference.
We test the arithmetic mean of recall values of ROUGE-1, ROUGE-2 and ROUGE-L in our setting of sentence containment.

Content-words: (Lebanoff, K. Song, et al. 2019) use content words to compare sentences. They define them as tokens that are neither stopwords nor punctuation. We use this definition in a bag-of-words approach. We compute the containment score of a source sentence in a target segment as follows: Let w_c be the common content words between source sentence and target segment and w_t the content words of the target segment, then the score is $\frac{w_c}{w_t}$.

Sentence-embeddings Furthermore we experiment with BERT-sentence embeddings (Devlin et al. 2019). We compute the cosine similarity between a sentence and a text, whereas the text can be a single sentence or sequence of sentences.

Further metrics: We also experimented with other metrics, like ROUGE-1-recall and word/lemma-containment. These are very similar to content words described above and have nearly the same scores. Specific use of named entities did not gave us improvement. We also experimented with more elaborate approaches. (Dias 2007) do not use a bag of words approach but exclusive links between words of source and target. Thus a word in the target can only be referenced by one word in the source, even if there is another match. Since they only establish links between lexical matching tokens we tried to adapt this concept on links that also represent semantic similarity. In order to keep the links exclusive, we view the matching possibilities as an assignment problem (Wikipedia 2021a). We employed an engine that can solve the matching problem, but we note that the computation effort is too high in the pipeline.

4.2.3.2 Holdout-set

The Holdout-set consists of sentence relations between individual sentences of a target article and one or more sentences of several source articles. We observed 25 source-target article pairs in Multi-News as well as in WCEP and checked 345 target sentences in total. We count 249 target-sentences having relations to 775 source sentences.

For Multi-News one assessor manually checked 25 randomly chosen source-target articles and identifies relations between source sentences and target sentences. To do so for a given target sentence the assessor decided for each sentence of a source article whether it is related or not. He sets a binary annotation, whereas *yes* indicating that the major content of a sentence is contained in the target sentence and *no* that the opposite is the case.

For WCEP we selected 25 article clusters with a short summary of at least two sentences. From each selected article cluster we ranked all articles of the cluster with at least 8 sentences by Jaccard Index with the summary and take the top 5 best scored articles. Following the assessor annotates all prepared source-target pairs as described for Multi-News. Table 8.1 in the appendix lists the detailed information about data preparation and annotation. As a result we get a gold reference for sentence relations between source sentences and single target sentence texts with in total 194 (Multi-News) and 55 (WCEP) mappings[6]. Table 8.2 in the appendix lists further statistics about the holdout-set. Table 4.7 gives an example of a mapping.

source sentences
• Eight of the gubernatorial seats up for grabs are now held by Democrats; three are in Republican hands.
• Republicans currently hold 29 governorships, Democrats have 20, and Rhode Island's Gov. Lincoln Chafee is an Independent.
• Polls and race analysts suggest that only three of tonight's contests are considered competitive, all in states where incumbent Democratic governors aren't running again: Montana, New Hampshire and Washington.
• While those state races remain too close to call, Republicans are expected to wrest the North Carolina governorship from Democratic control, and to easily win GOP-held seats in Utah, North Dakota and Indiana.
target sentence
• The GOP currently controls 29 of the country's top state offices; it's expected to keep the three Republican ones that are up for grabs (Utah, North Dakota, and Indiana), and wrest North Carolina from the Dems.

Table 4.7: Multi-News - annotation example

4.2.3.3 Extraction Experiments

The holdout-set allows us to compute precision, recall and f-measure. We test the previously described metrics, ROUGE-1-2-L, content-words and sentence-embeddings on a range of threshold values. For each reference pair of articles (source sentences) and summary (target text) in the holdout-set we use all the containment metrics paired with a threshold value to retrieve sentences from the articles for a single

[6]A mapping is the set of all relations between a single target text and respective source sentences.

target sentence. Following we compare the retrieved sentences with the respective annotated sentences of the reference. We assess recall, precision and the f-measure. These values are computed as follows: For a given target sentence, let s_c be the number of common sentences of reference and retrieved sentences, s_{ref} the number of reference sentences and s_m the number of retrieved sentences by the metric, then is recall $= \frac{s_c}{s_{ref}}$, precision $= \frac{s_c}{s_m}$ and f-measure $= \frac{2*recall*precision}{recall+precision}$. We compute precision, recall and f-measure for each target sentence take the average of the total holdout-set.

Figures 4.2 and 4.3 show precision, recall and f-measure separated for Multi-News and WCEP holdout-set[7]. In both scenarios the content-word-containment metric achieves the best results regarding the f-measure. We decide to choose the content-word metric with a threshold of 0.5 because of the high precision. Since we have a lot of input data, we can compensate the loss in recall.

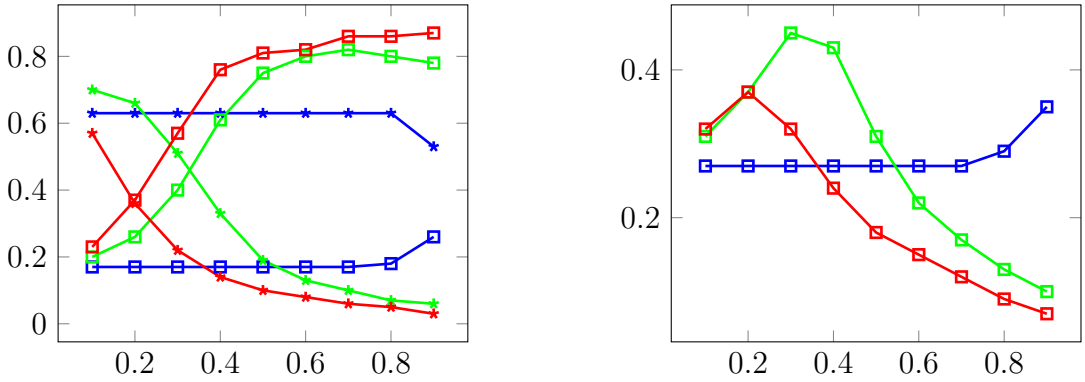

Figure 4.2: Multi-News - extraction experiments results; x-axis: threshold; y-axis: left - precision (square) and recall (star); right - f-meausre; blue: sentence-embeddings, green: content-words, red: ROUGE-1-2-L

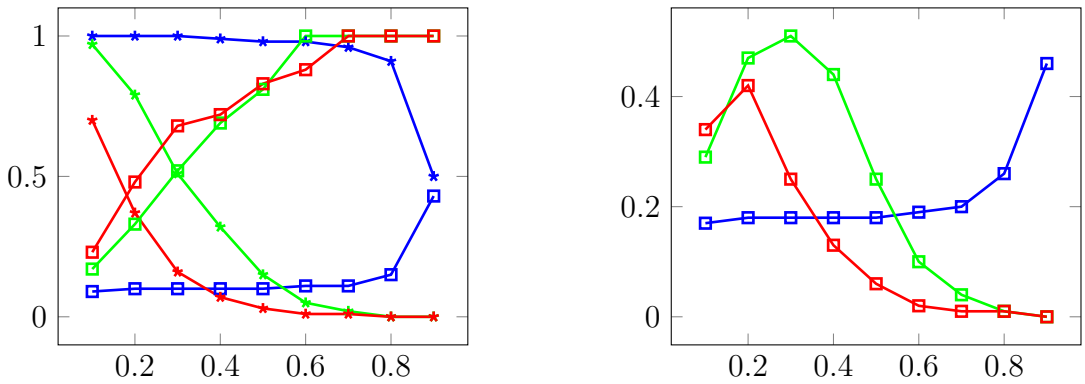

Figure 4.3: WCEP - extraction experiments results; same legend as in figure 4.2

[7]The exact values can be found in the appendix in table 8.3

4.2.4 Selection

In the final stage, given a cohesive text segment and the corresponding related source sentences, our algorithm selects a well-suited set among the related source sentences. This selection process aims for fulfilling the criteria of a valid training pair (see section 3.1). To achieve this we use the same method as proposed in (Lebanoff, K. Song, et al. 2019). Afterward we filter all insufficient training pairs by an additional check. Code-block 3 describes our approach.

Algorithm 3 select-sentences

```
1:  Input
2:      check-func                  ▷ Validation function passed as parameter
3:      max-n-sents          ▷ configuration value (see Limitations search space)
4:      min-n-sents
5:      segment                                          ▷ cohesive anchor text
6:      related-sents          ▷ retrieved sentences from extraction stage
7:  Output
8:      selected-sents
9:  procedure SELECT-SENTENCES(metric, t, segment, related-sents)
10:     selected-sents ← sentence-ranking(segment,related-sents,max-n-sents)
11:     n-selected-sents ← number of selected-sents
12:     if (n-selected-sents ≥ min-n-sents) and
13:         check-func(selected-sents) then
14:         return selected-sents
15:     else
16:         return null
17:     end if
18: end procedure
```

In the remaining part of the chapter we motivate and describe the selection method, the configuration min/max-n-sents and the check-function.

4.2.4.1 Limitations search space (min/max-n-sents)

The arguments min-n-sents and max-n-sents limit the space of valid training pairs. We only consider training-pairs with a number of source sentences in between those two values. The values of min/max-n-sents depend on the number of sentences in the segment. Let n be the number of sentences in the segment, then we define min-n-sents := n+1 to foster the operation of sentence fusion. In order to allow input sentences with information redundancies and fusion pairs that have more extensive transformation involved, we set the upper bound max-n-sents to 2n+1. In this range of input sentences, fusion pairs with all kind of operations can exist.

4.2.4.2 Sentence ranking

Since the related sentences were individually chosen, but not in regard to the other retrieved sentences, we still need to seek a subset among them, that fulfills the requirements of the source sentences (see section 3.1). We use the technique of *sentence ranking* that we already described in the section 3.2:
In the algorithm 3 (line 10) the ranking-mechanism is applied to sort the related-sentences passed to the selection stage. In addition to the two-word-overlap constraint, the iterative ranking process stops if a certain number of sentences is reached. Then the ranked-sentences get returned. We refer to this number as max-n-sents, described in *Limitations search space*.
This process outputs a subset of the related sentences that covers as much information of the target segment as possible. In addition redundant information among source sentences that also appears in the target gets minimized, but not prevented. This optimization is designed to create examples that fit the necessary criteria of fusion pairs.

We also experimented with other approaches. Once all the related-sentences from the extraction stage are retrieved, various subsets $\subset$ related-sentences can suit the criteria. A brute force approach could be implemented with checking each possible subset within the *limitations search space* through the validation metric (see section 4.2.4.3). However it is not feasible to do this, because the resulting number of subsets is too huge: Let n be the number of related-sentences, f the min-n-sents value and c the max-n-sents value, then the number of subsets is $\sum_{k\in\{f,\dots,c\}} \frac{n!}{(n-k)!}$.
Therefore we dynamically constructed a tree-graph beginning from the root. We employed a gain metric that decided which nodes to put in next. With this technique we were able to traverse the tree-graph in an efficient order. Any gain-metric can be equipped, what allows to optimize regarding all kind of properties of the fusion pairs. Nevertheless the implementation was far slower than that one of sentence-ranking and does not give any benefits. Therefore we discarded this approach and continued with (Lebanoff, K. Song, et al. 2019)'s solution. Maybe this idea can be used if the criteria to the subsets become more complex.

4.2.4.3 Sentence selection - validation metric

After the related-sentences get ranked and selected and pass the min-n-sents constraint we additionally validate the outcome with a specific function, here called check-func (see algorithm 3, line 13).

Up to this point each source sentence is related to the target and the sentence ranking method returns a subset of the related sentences that is optimized regarding the information coverage of the target segment. It is still necessary to validate whether the target coverage is on a sufficient level and furthermore if the information coverage is distributed over the individual target sentences rather than just concentrated in a certain part of the target segment.
We propose the following metric: First, we calculate the content-word-containment score for each individual sentence in the cohesive segment (here: S_{tgt}). In order to do so we handle the compound of selected source sentences (here: S_{src}) as one text. We then use the containment-metric (here: f^c) to calculate the score of each individual target sentence of the segment in the compound of source sentences. We finally select the minimum containment score. We refer to this metric as min-containment:

$$\text{min-containment}(S_{src}, S_{tgt}) := \min\{f^c(s_{tgt}, S_{src}) | s_{tgt} \in S_{tgt}\}$$

Afterward we compare the returned score against a threshold value to validate the selected-source-sentences. The min-containment metric returns the minimal coverage score of an individual target sentence in the compound of source sentences. We utilize the previously described content-word metric that is also applied in the extraction stage. The benefit of taking the minimal score rather than computing the containment of the target-sentences as a whole, is that we prevent an unbalanced distribution of covered information in the target. In the latter case the by-source-sentences-covered information could be concentrated in one part of the target segment while in the remaining target-sentences no information is covered. This could still result in a sufficient score, that would pass the threshold, but does not meet the requirements of our fusion pairs.

4.2.4.4 Validation metric - threshold experiment

The min-containment function explained in the section before returns a score between 0 and 1. We use this score and compare it against a threshold value to validate the selected-sentences. To determine this score we conduct an experiment: We test with a sample of 100 instances of Multi-News and WCEP and pass the units to the pipeline. We perform all stages until the selected-sentences reach the check-func in the code excerpt above (see algorithm 3, line 13). Instead of filtering selected-sentences we assign labels to them. Each label indicates whether the unit would pass the check-function with a certain threshold. Afterward we compute ROUGE-1, ROUGE-2 and ROUGE-L f-measures between the compound of source

threshold	#output-units	rouge-1-f	rouge-2-f	rouge-l-f	rouge-mean	annotation (good/ medium/ bad)
0.3	66	0.52	0.25	0.32	0.36	
0.4	51	0.55	0.29	0.34	0.39	
0.5	35	0.58	0.33	0.38	0.43	
0.6	28	**0.59**	**0.35**	**0.39**	**0.45**	22 / 4 / 2
0.7	17	**0.62**	**0.41**	**0.44**	**0.49**	14 / 3 / 0

Table 4.8: Selection Threshold Experiment - Results

sentences and the target segment of each resulting pair. We compute the average scores for each threshold-label. In addition we manually annotate examples with one of three classes: *good* - the fusion pair requirements are fulfilled, *medium* - not all fusion pair requirements are fulfilled, but the example seems bearable and *bad* - the example is not sufficient.
Let $t_1, ..., t_n$ be threshold values in strict order $t_1 < \ldots < t_n$, then the fusion pairs with label t_j will also have label t_{j-1} for $j \neq 1$. Therefore we have following order of threshold-labeled sentences:

$$\{\text{pairs with label } t_n\} \subset \{\text{pairs with label } t_{n-1}\} \subset \ldots \subset \{\text{pairs with label } t_1\}$$

Since we believe that with a descending threshold value the quality of the fusion pairs regarding their requirements decreases, we began annotating the examples with high threshold values. We stopped when the mean quality of examples was not sufficient anymore. Table 4.8 presents the results. We stopped annotating examples at a threshold of 0.6 and decided on a threshold of 0.7 for min-containment validation.

4.2.5 Corpus Processing

Finally we execute the pipeline on the preprocessed Multi-News and WCEP corpora. Therefore we utilized an 64 core machine and parallelized the execution. The detection stage (first stage) does not require much computation time, since it only processes the sentences of one article. The extraction and selection stage deal with sentences of more than one article, in case of WCEP even with up to hundred articles. To cope these huge amounts of data, we tackle the extraction and selection of source sentences of one cohesive segment as one job. We bundle these jobs in equally sized batches and distribute the processing of these batches over the 64 cores. With this approach we achieve a good capacity utilisation.

We ran the pipeline on the complete Multi-News corpus, with a processing time of about 1,5 days. For WCEP we only processed the first 1250 article-clusters with a processing time of approximately 7 days.

4.3 Corpus Statistics

4.3.1 Distribution of fusion pairs

This section provides details about the generated fusion corpora. Table 4.9 and 4.11 count the number of occurring fusion pairs with different numbers of source and target sentences. As we can see do many combinations of length in source and target sentences exist. This means our dataset includes many sentence fusion cases with varying numbers of source and target sentences. The target text length is nearly equally distributed, the number of source sentences slightly fluctuates, especially fusion cases with ten sentences in the input are less represented by the datasets. All in all we do not think that further statistical adaptions regarding the distribution of fusion cases (e.g. down-sampling) are necessary.

4.3.2 Content Containment Scores

To get a better understanding of the created examples we capture the coverage scores, using our containment metric, between source and target of fusion pairs. We check the containment of the bundled source sentences in the target text and vice-versa, the coverage of the target text in the source sentences. We categorize each example by its number of sentences in the target text and separately compute the average containment score for each category. Table 4.10 and 4.12 present the results.

The containment of source sentences in the target is decreasing in both corpora with rising number of target sentences. It even drops below the extraction threshold value of 0.5. Since the fusion pairs with more target sentences also have more source sentences (see tables 4.9 and 4.11) the number of non-covered content words of the source increases as well. Covered content words often redundantly appear in multiple sentences, because their source sentences are all related to the same target. Hence the portion of covered and non-covered words shrinks as we can see in the tables 4.10 and 4.12. This could mean that in longer input texts our datasets has a stronger focus on merging common parts of the source in the target.

The containment score of the target text in source sentences does not strikingly fluctuates as strong as above and is close to the selection threshold.

number of target sentences		1	2	3	4	5	total
frequency		12980	11877	10446	8968	7494	51765
relative		0.25	0.23	0.2	0.17	0.14	
distribution of mapping sizes (number of source sentences)	2	9731					9731
	3	3249	4784				8033
	4		3634	2174			5808
	5		3459	2520	1078		7057
	6			2152	1450	512	4114
	7			3600	1580	792	5972
	8				1472	1031	2503
	9				3388	1055	4443
	10					960	960
	11					3144	3144

Table 4.9: Multi-News - Distribution fusion pairs

n-tgt-sents	src-tgt-containment	tgt-src-containment
1	0.57	0.72
2	0.54	0.71
3	0.52	0.7
4	0.5	0.69
5	0.48	0.69

Table 4.10: Multi-News - containment scores

number of target sentences		1	2	3	4	5	total
frequency		81272	88821	84225	77951	71424	403693
relative		0.2	0.22	0.2	0.19	0.18	
distribution of mapping sizes (number of source sentences)	2	51639					51639
	3	29633	26425				56058
	4		25482	12359			37841
	5		36914	16606	5766		59286
	6			16177	9536	2981	28694
	7			39083	10996	5111	55190
	8				11082	6701	17783
	9				40571	7732	48303
	10					7880	7880
	11					41019	41019

Table 4.11: WCEP - Distribution fusion pairs

n-tgt-sents	src-tgt-containment	tgt-src-containment
1	0.55	0.78
2	0.53	0.78
3	0.51	0.77
4	0.49	0.77
5	0.47	0.76

Table 4.12: WCEP - containment scores

Chapter 5

Cohesive Text Generation

This chapter deals with the sequence-to-sequence model that we employ to learn the extended sentence fusion task. We first describe the architecture of the underlying Transformer model. We then explain the procedure of pretraining and the pretrained BART, a powerful language model. Finally we finetune BART on our sentence fusion task.

5.1 Transformer-based Sequence to Sequence Models

The Transformer model architecture (Vaswani et al. 2017) was originally used for machine translation. It is very successful and is used more frequently than other recurrent architectures, like LSTMs in (Peters et al. 2018). It consists of an encoder and a decoder. The encoder transforms a sequence of input symbols to a sequence of internal representations. The latter one is passed to the auto-regressive decoder block which generates an output sequence by consuming the previously generated output positions.
Both encoder and decoder are composed of multiple layers including stacked self-attention. The self-attention mechanism measures the relationship for each word in the sequence to other words in the sequence. This allows efficient word encodings with contextual information, even if the dependent words have a high distance in between.

Pretraining: The term *pretraining* refers to the process of training a neural network on a general task. The subsequent *finetuning* is the specialisation of the pretrained model on a certain end task, also called downstream task. Works have shown that pretraining a model can improve the overall performance on various

NLP tasks, like sentiment analysis (Dai and Le 2015), text classification (Howard and Ruder 2018a) or Question answering and commonsense reasoning (Howard and Ruder 2018b). In the pretraining phase the model acquires understanding of the language through extensive unsupervised learning of a certain task, like Masked Language Modeling or Next Sentence Prediction (Devlin et al. 2019) on huge datasets[1]. Afterwards the model can be specialized on a downstream-task. The finetuning strategy slightly adapts all parameters learned in the pretraining phase to perform on the downstream task, but requires less specific end-task data and compute resources to achieve remarkable results (Devlin et al. 2019).

5.2 BART

BART (short for Bidirectional Auto-Regressive Transformer) is a pretrained language model that can be applied to many NLP downstream Tasks, in particular to Natural Language Generation and Summarization (Lewis et al. 2019). Since the extended sentence fusion task is similar to Summarization we use the same finetuning procedure to optimize the model on our task.

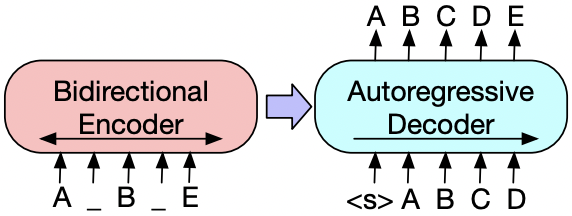

Figure 5.1: BART - architecture (image source: (Lewis et al. 2019))

BART relies on the Transformer model architecture. The Encoder and Decoder were both independently utilized for unsupervised pretraining: BERT applies the Encoder in a pretraining setting with general learning objectives. Through that process it acquires language representations that contains information about the whole context, not just the previous tokens. Therefore the encodings are called bidirectional (comp. Devlin et al. 2019). In contrast the GPT approach uses the transformer Decoder to auto-regressively predict the next tokens. Therefore it can be used for Language Generation Tasks. In the pretraining it only uses the previously appearing tokens and does not incorporate contextual information from both directions (comp. Howard and Ruder 2018b).

[1]In case of GPT-3, one of the biggest language models, the training-set consists of about 500 billion tokens (Brown et al. 2020).

BART uses the bidirectional Encoder as well as the auto-regressive Decoder. In the pretraining phase the model is optimized to reconstruct the original document in an auto-regressive fashion. As input the encoder is supplied with a corrupted document, e.g. text spans are replaced with mask symbols.

BART achieved similar results as comparable bidirectional encoders (comp. Liu et al. 2019) and is capable of language generation, like summarization. (Lewis et al. 2019) experimented with two version of the BART model that differ in the number of layers in the Encoder and Decoder block. Other works have shown that large scale pretraining significantly improves the performance on downstream tasks (Yang et al. 2019),(Lewis et al. 2019). Thus the larger BART model was pretrained on "160Gb of news, books, stories, and web text." Afterwards (Lewis et al. 2019) finetuned the large model on the CNN/DM news summarization dataset (Hermann et al. 2015). The resulting model outperforms many other models on this task. We refer to this finetuned model as *BART-Large-CNN*.

Finetuning on sentence fusion: In order to optimize the BART-Large-CNN on the extended sentence fusion task we finetune it again on the sentence fusion corpora. We train two models independently on Multi-News and on WCEP fusion pairs. We split both fusion pair datasets into train/val/test with a 80/10/10 distribution. Afterward we employ the same finetuning scripts used for summarization[2]. We executed the scripts on a GPU enabled machine. The finetuning procedure of the Multi-News model happened on 40 000 training pairs within 2.5 hours. In case of WCEP we had 320 000 training pairs and it took about one day.

[2]We utilized the finetuning scripts proposed in the Huggingface example repository (Huggingface 2021).

Chapter 6

Evaluation

In order to survey the quality of the dataset and the performance of our trained model we conduct a manual and automatic evaluation. In the following part we explain our setup for both experiments and interpret the results.

6.1 Automatic Evaluation

ROUGE: To quickly get a better understanding of the data we computed several metrics on the fusion pair corpus and the model prediction results. So that we can quickly see the results of the training process, we compared the reference target-text with generated text of the model. To do so we calculate the ROUGE-1,ROUGE-2 and ROUGE-L f-measures between reference and model output of records of the training-splits for both corpora, Multi-News and WCEP.

We labeled each record of the training-splits with the number of sentences in the target reference. Following we created bins for each label. For each of these bins we separately compute the metrics. Table 6.1 and 6.2 present the results. Another approach would be to differentiate by the number of input sentences. We preferred the first one, because there are only 5 bins and additionally is the number of target sentences also indicating the amount of content, whereas source compounds with the same number of sentences can have different amounts of content, because of redundancies.

n-tgt-sents	ROUGE-1	ROUGE-2	ROUGE-L
1	0.61	0.4	0.48
2	0.58	0.32	0.38
3	0.57	0.29	0.33
4	0.57	0.26	0.30
5	0.57	0.25	0.28

Table 6.1: Automatic Evaluation - Multi-News - ROUGE

n-tgt-sents	ROUGE-1	ROUGE-2	ROUGE-L
1	0.57	0.35	0.46
2	0.54	0.29	0.37
3	0.53	0.26	0.32
4	0.52	0.23	0.28
5	0.52	0.22	0.27

Table 6.2: Automatic Evaluation - WCEP - ROUGE

As we can see is the f-measure in all cases relatively high. The f-measure is a combination of recall and precision[1]. If the resulting f1 score is high, then both, precision and recall are high as well; if it is low then recall or precision or both are low. Furthermore we can see that the metric scores are descending with increasing number of target sentences, where as the ROUGE-1 score is just minimally decreasing, but ROUGE-2 and ROUGE-L show striking differences in varying number of target sentences.

We conclude that the fine-tuning process worked quite well and the model performs good on the test examples. We want to emphasize here that the examples were generated by our pipeline. Since the pipeline could produce deficient source target pairs regarding our definition of the sentence fusion task, the results of the automatic evaluation only refer to the performance on the problem determined by the generated dataset. Additionally we see that with increasing number of target sentences the model output and the reference become more different. To evaluate the model on the actual sentence fusion task we conduct an extensive manual evaluation (see chapter 6.2).

[1] f-measure without weight: $\frac{2*precision*recall}{precision+recall}$

n-tgt-sents	src-in-ref	src-in-gen	ref-in-src	gen-in-src
1	0.57	**0.68**	0.72	**0.82**
2	0.54	**0.63**	0.71	**0.79**
3	0.52	**0.58**	0.7	**0.78**
4	0.5	**0.56**	0.69	**0.78**
5	0.48	**0.54**	0.69	**0.78**

Table 6.3: Automatic Evaluation - Multi-News - Containment

n-tgt-sents	src-in-ref	src-in-gen	ref-in-src	gen-in-src
1	**0.55**	0.53	**0.78**	0.73
2	**0.53**	0.53	**0.78**	0.73
3	**0.51**	0.5	**0.77**	0.72
4	**0.49**	0.47	**0.77**	0.71
5	**0.47**	0.44	**0.76**	0.70

Table 6.4: Automatic Evaluation - WCEP - Containment

Containment scores: Furthermore we calculated the containment-metric[2] on the model output. We compute the containment scores for both directions: source sentences contained in the model output and vice-versa. We use the same labels from above and bin the examples by their number of target sentences. For each of these bins we separately compute the metrics.

As we can see in table[3] 6.3 and 6.4, the metric scores are descending with increasing number of target sentences. We compared the values with the results from the corpus statistics (we already computed the same metrics for training examples, see section 4.3). In the case of Multi-News the model output has higher containment scores than the pipeline generated examples. Interestingly the opposite is the case for WCEP. Here the pipeline generated examples have higher containment values. We conclude that the model properly processes the input sentences and that the coverage of content is similar distributed as in the training data.

[2]we use the same containment-metric as in the extraction stage; comp. section 4.2.3.1 - content words

[3]src: source compound, ref: reference target-text, gen: generated model text

6.2 Manual Evaluation

6.2.1 Procedure/Setup of the Assessment

In the manual evaluation we simultaneously review training pairs and model outputs. We examine samples from test splits of both fusion-pair corpora and annotate the examples according to an evaluation guide: we observe 4 categories: Omission, Inaccuracy Intrinsic, Inaccuracy Extrinsic and Structure & Coherence. Table 6.6 explains the categories and the labels that get assigned to each record.

For both fusion pair corpora, 50 examples were randomly selected with 10 records per occurrence of 1,2,...,5 sentences in the target segment. Following we applied the model on the source compound of the records. The assessor can evaluate training example quality and the model quality in one experiment. She only has to read the source sentences of one example to evaluate both scenarios.

One assessor checked 200 records (two scenarios - corpus and model performance - and two corpora - Multi-News and WCEP- with each 50 examples). After the assessment we analyze the results in detail and give several examples that show the performance of the pipeline and the trained model. The assessed records can be analyzed along 4 dimensions: training pairs and model output (scenario), The number of target sentences, Multi-News and WCEP (corpus) and the kind of error/quality and its severity. The next sections present our results.

We were guided by the evaluation setup of (Huang et al. 2020), who assessed the quality of text summarization systems.

Kind of Error Quality	**Description**	**Severity**	**Description**
Omission	This error indicates how much of the important information is missing.	No	No information is missing.
		Low	Less Information is missing: adjectives, second tier nouns, verbs, etc. - the output still conveys all of the main content.
		Moderate	Clearly a part of the content is missing, but in favor of coherence the omission is bearable.
		High	Clearly a part of the content is missing.
Inaccuracy Intrinsic	Is information from the source wrongly modified?	No	No information is wrongly modified.
		Minor	The proportion of wrongly modified information impacts only small, minor part of the output.
		Critical	The wrongly modified information affects bigger parts of the output and results in non bearable change of meaning.
Inaccuracy Extrinsic	Is new Information added in the output?	No	No new information is added.
		Verifiable	The added information can be verified via websearch or other sources or assessed from the context of the input sentences as relevant.
		Not verifiable	The added information neither can be verified via websearch or other sources nor assessed from the context of the input sentences as relevant.
		Incorrect	The added information can be assessed as factually incorrect after websearch or use of other sources.
Structure and Coherence	How well-structured is the output?	Very Good	Range of statements about linguistic quality of the output text with gradation in Structure and coherence.
		Good	
		Barely Acceptable	
		Poor	
		Very Poor	

Table 6.6: Manual Evaluation - annotation guide - description of labels

6.2.2 Corpus: Error Analysis

In this part of the analysis we examine the errors/quality of the training-pairs independently for each fusion pair corpus.

Omission: As we can see in table 6.7 is the omission in both corpora quite high. 54 of 100 records are labeled with high omission of information from source sentences in the target segment. Table 6.8 indicates that the omission level increases with the number of sentences in the target segment. The latter effect could be explained with the rising inaccuracy of the content-word metric used in the extraction stage for longer text. Since this metric is based on the bag of words approach, it is more accurate if the words only appear in individual sentences. It is more likely that multiple words of a single sentence refer to the same context, than multiple words in more than one sentence do. The general high omission rate could be encountered with a higher threshold in the extraction stage. The utilized holdout set in the extraction stage only uses binary annotation to indicate that the major part of the sentence is contained in a single sentence. Such a relation still allows further parts missing in the target. Since the containment task is different regarding varying number of sentences in the target the holdout-set is less meaningful for the containment task with multiple target-sentences. Figure 6.2 gives an example with a high level of omission in the target-text and figure 6.1 gives an example without omission in the target-text

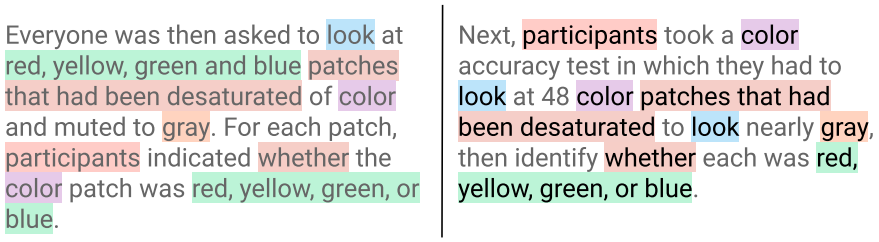

Figure 6.1: Example - corpus - WCEP - Omission - No

corpus	No	Low	Moderate	High
Multi-News	3	8	16	23
WCEP	1	9	9	31
both	4	17	25	54

Table 6.7: Manual Evaluation - corpus - Omission

According to Ni, China's recent military displays are part of a wider strategy aimed at intimidation and deterrence, though he says it is unlikely that China will actually intervene at this stage. Both Hong Kong and Macao are former European colonies that were returned to China in the late 1990s. "China is doing more and more to pressure Hong Kong, its people and its organisations."

Though China has become more vocal about its ability to mobilize armed forces in Hong Kong, experts say the military displays are part of a wider strategy aimed at intimidation and deterrence, and it is unlikely that China will actually intervene at this stage.

Figure 6.2: Example - corpus - WCEP - Omission - High

n-tgt-sents	No	Low	Moderate	High
1	2	2	6	10
2	1	7	4	8
3	1	4	6	9
4	0	4	4	12
5	0	0	5	15

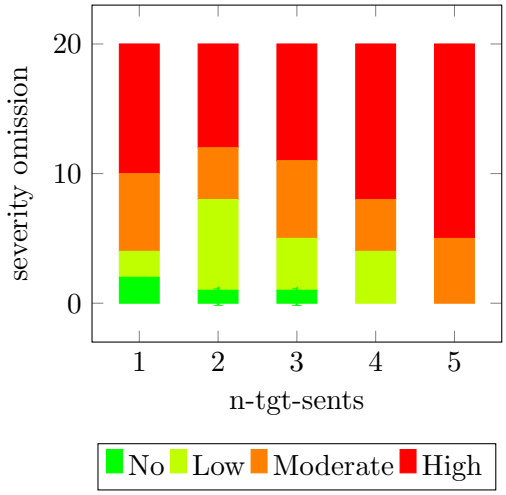

Table 6.8: Manual Evaluation - corpus - Omission by n-tgt-sents

Inaccuracy Intrinsic: This error is in both corpora relatively low in all but a few cases (see table 6.9). In case of Multi-News the target text segments are extracted from human written summaries. These summaries provide the relevant information of the source articles and contain less mistakes. Our pipeline could extract related parts for summary and source articles without many mistakes. Even for WCEP the generation approach keeps the inaccuracy error low. The original clusters of the WCEP corpus contain articles about the same topic. In theory our approach works well as long as the articles contain less contradictory information. Also here our pipeline is able to process target article and source articles without many mistakes. The Inaccuracy Error is low for all numbers of target sentence as we can see in table 6.10. Figure 6.3 shows an example with critical erros, including many problems with numbers.

corpus	No	Minor	Critical
Multi-News	48	2	0
WCEP	46	2	2
both	94	4	2

Table 6.9: Manual Evaluation - Inaccuracy Intrinsic

n-tgt-sents	No	Minor	Critical
1	18	1	1
2	20	0	0
3	19	0	1
4	19	1	0
5	18	2	0

Table 6.10: Manual Evaluation - Inaccuracy Intrinsic by n-tgt-sents

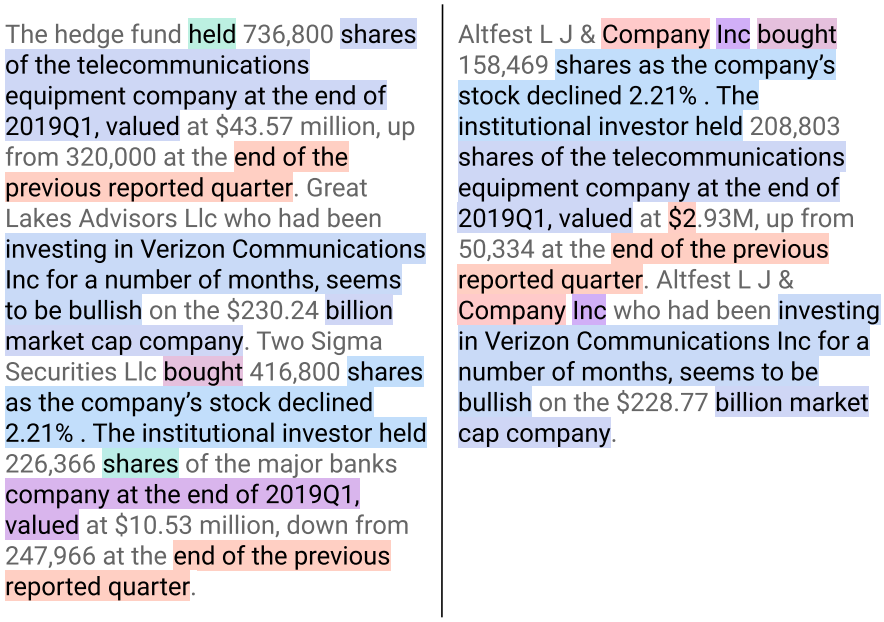

Figure 6.3: Example - corpus - WCEP - Inaccuracy Intrinsic - Critical

corpus	No	Verifiable	Not verifiable	Incorrect
Multi-News	23	27	0	0
WCEP	31	19	0	0
both	54	46	0	0

Table 6.11: Manual Evaluation - corpus - Inaccuracy Extrinsic

Inaccuracy Extrinsic: Table 6.12 shows that 46 of 100 examined records add new Information to the target segment. Since the target segment is extracted from an human written article in the web we can see all new information as verifiable. The effect of adding information seems independent to the number of target

n-tgt-sents	No	Verifiable	Not verifiable	Incorrect
1	10	10	0	0
2	11	9	0	0
3	12	8	0	0
4	13	7	0	0
5	8	12	0	0

Table 6.12: Manual Evaluation - corpus - Inaccuracy Extrinsic by n-tgt-sents

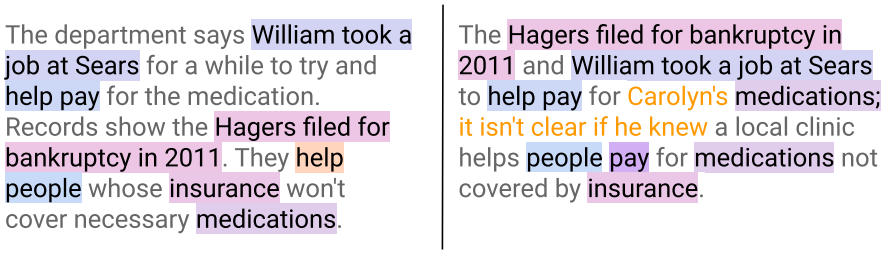

Figure 6.4: Example - corpus - Multi-News - Inaccuracy Extrinsic - Verifiable

sentences in the segment (comp. table 6.12). In order to decrease the added information we could increase the threshold in the selections stage. Figure 6.4 gives an example for Inaccuracy Extrinsic.

Structure and Coherence: We observe that the linguistic quality as well as the coherence of the target segments is good (comp. table 6.13). That confirms our hypothesis to find coherent text segments in a closed text (see section 3.2. Furthermore we can see that the simple technique (see section 4.2.2) to use consecutive sentences as target segments works properly in the majority of cases. This is true for single sentences as well as for segments with more sentences (see table 6.14). Figure 6.5 shows an problematic example.

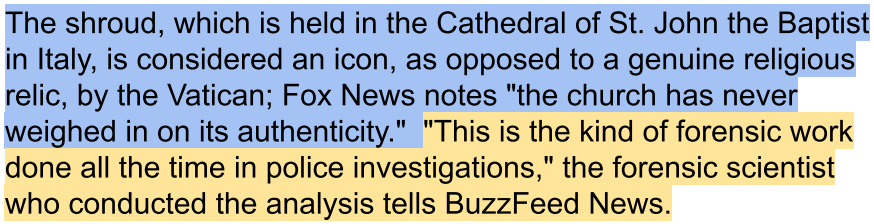

Figure 6.5: Example - corpus - Multi-News - Structure and Coherence - Barely Acceptable

corpus	Very Good	Good	Barely Acceptable	Poor	Very Poor
Multi-News	39	9	2	0	0
WCEP	39	8	3	0	0

Table 6.13: Manual Evaluation - corpus - Structure and Coherence

n-tgt-sents	Very Good	Good	Barely Acceptable	Poor	Very Poor
1	18	2	0	0	0
2	16	3	1	0	0
3	14	5	1	0	0
4	14	5	1	0	0
5	16	2	2	0	0

Table 6.14: Manual Evaluation - corpus - Structure and Coherence by n-tgt-sents

6.2.3 Model: Error Analysis

In this part of the evaluation we examine the model performance. We utilize the same criteria as in the training pairs analysis. In addition we compare the results of the training pair evaluation and the model performance to see whether the training process has impact on the quality. We want to point out that we have two models, one trained on Multi-News fusion-pairs, the other trained the WCEP fusion-pairs. We use the test split of Multi-News fusion-pairs to evaluate the Multi-News model, analog for WCEP. We gather the results of the assessment from both fusion pair corpora to make conclusions and handle them separately when needed. If no distinction is made between the corpora in the following tables, then labeled records from both are combined.

Omission: Table 6.15 shows that 37 of 100 examined records contain a high level of omission. Compared with the training data this error is far lower (see table 6.16). We can see that the training process has positive influence on the intake of information from the source sentences in the generated target segment. As well as in the training pairs, we can observe, that an increase of information loss comes along with a higher number of target sentences in the output (see table 6.17).

corpus	No	Low	Moderate	High
Multi-News	6	18	13	13
WCEP	4	9	13	24
both	10	27	26	37

Table 6.15: Manual Evaluation - model - Omission

Scenario	No	Low	Moderate	High
corpus	4	17	25	54
model	10	27	26	37

Table 6.16: Manual Evaluation - comparison (corpus/model) - Omission

l"Kate Spade has a truly unique and differentiated brand positioning with a broad lifestyle assortment and strong awareness among consumers, especially millennials," Coach CEO Victor Luis said in a statement. That's a 9 percent premium to its Friday closing price of $16.97. " Coach will pay $18.50 per share of Kate Spade & Company.	In a statement, Coach CEO Victor Luis praised Kate Spade's "unique and differentiated brand positioning with a broad lifestyle assortment and strong awareness among consumers, especially millennials." At $18.50 per share, Kate's shares were trading at a 9% premium over Friday's closing price.

Figure 6.6: Example - Multi-News - model - Omission - Low

n-tgt-sents	No	Low	Moderate	High
1	6	4	8	2
2	3	8	2	7
3	1	8	6	5
4	0	4	8	8
5	0	3	2	15

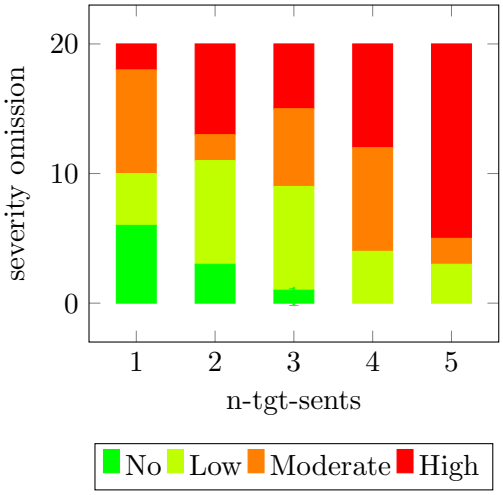

Table 6.17: Manual Evaluation - model - Omission by n-tgt-sents

Inaccuracy Intrinsic: The examined sample show a high portion of critical inaccuracy: 46 of 100 records contain inaccuracy errors (see table 6.18). Many times parts of the sentences get stitched together that are not to be connected. This shifts the meaning of the statements conveyed in the source sentences. Table 6.20 indicates that this phenomenon happens more often in longer target segments with more sentences. Table 6.19 shows that this error is a result of the generation technique of the model and is not part of the training data - the sample of the training pairs does not contain this issue. Figure 6.7 shows a critical example.
Furthermore the model tends to insert random news provider. This is a result of the domain of the initial datasets Multi-News and WCEP. Articles from both datasets often contain information about the news-provider of the article. Because the same information can be provided by multiple news-sources, sentences exist with the same content but different news-providers. So it can happen that a wrong news provider gets inserted in the output or a completely new one. Because this seems to be a structural problem the assessor did not weigh this error strongly.

corpus	No	Minor	Critical
Multi-News	24	7	19
WCEP	19	4	27
both	43	11	46

Table 6.18: Manual Evaluation - model - Inaccuracy Intrinsic

Scenario	No	Minor	Critical
corpus	94	4	2
model	43	11	46

Table 6.19: Manual Evaluation - comparison (corpus/model) - Inaccuracy Intrinsic

n-tgt-sents	No	Minor	Critical
1	10	2	8
2	13	0	7
3	9	4	7
4	6	4	10
5	5	1	14

Table 6.20: Manual Evaluation - model - Inaccuracy Extrinsic by n-tgt-sents

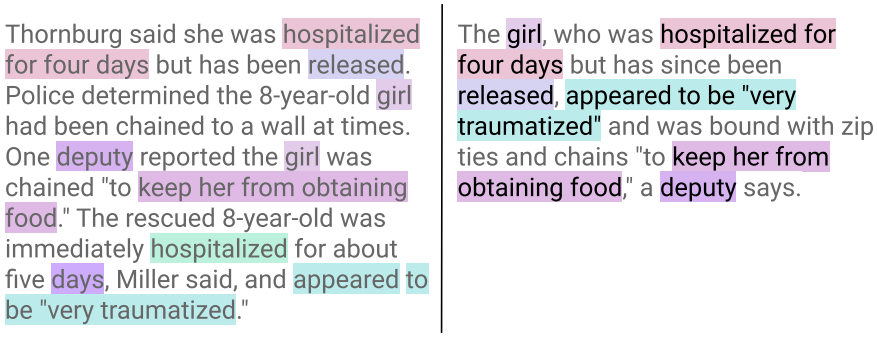

Figure 6.7: Example - corpus - wcep - Inaccuracy intrinsic - critical

Inaccuracy Extrinsic: As table 6.21 reveals, the model adds new information in 44 of 100 cases. Cause of the pre-training the model is enriched with information from many other text documents. Therefore it can add true content, as verified in 12 records, but in 30 cases it is not verifiable. In isolated cases the new content is proven to be incorrect. This phenomenon seems to be independent from the number of target sentences (comp. table 6.22). Table 6.23 shows that the model adds new content as frequently as this happens in the training examples. This means that the model adapts this bad property from the training data, which could perhaps be avoided with better training sources (comp. section 6.2.2). Figure 6.8 gives an example, where new true content is added.

corpus	No	Verifiable	Not verifiable	Incorrect
Mutli-News	36	3	10	1
WCEP	20	9	20	1
both	56	12	30	2

Table 6.21: Manual Evaluation - model - Inaccuracy Extrinsic

n-tgt-sents	No	Verifiable	Not verifiable	Incorrect
1	13	1	6	0
2	8	3	9	0
3	14	4	1	1
4	13	1	5	1
5	8	3	9	0

Table 6.22: Manual Evaluation - model - Inaccuracy Extrinsic by n-tgt-sents

Scenario	No	Verifiable	Not verifiable	Incorrect
corpus	54	46	0	0
model	56	12	30	2

Table 6.23: Manual Evaluation - comparison (corpus/model) - Inaccuracy Extrinsic

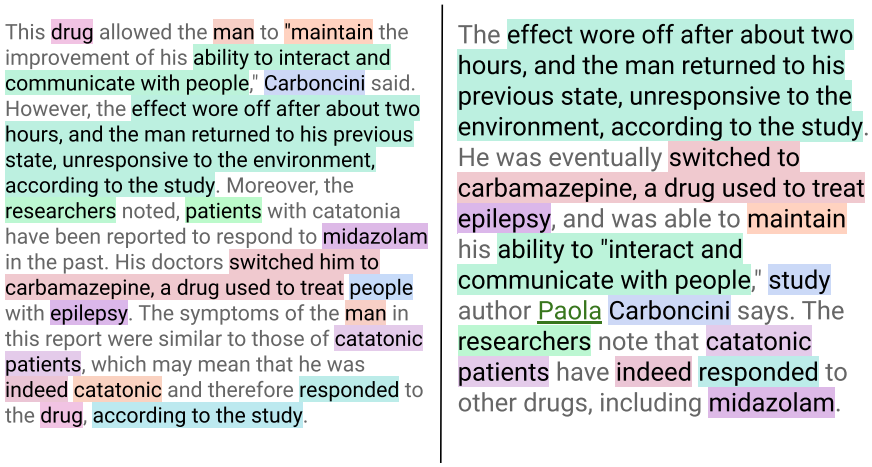

Figure 6.8: Example - model - Multi-News - Inaccuracy extrinsic - Verifiable

Structure and Coherence: In all but a few cases (95/100) the linguistic quality as well as the coherence of target segments is very good or good (see table 6.24). This result proves that the trained model is able to connect sentences in a coherent and cohesive matter. As other works have shown (see section 5.1), has the pre-training massive benefits on the linguistic quality of the downstream task, as we can also see in our results. Tables 6.25 and 6.26 are included for completeness and show structure & coherence with regard to the number of target sentences and in comparison with the training data. In figure 6.9 is a problematic case with time coherence issues shown.

corpus	Very Good	Good	Barely Acceptable	Poor	Very Poor
multiNews	41	7	2	0	0
wcep	37	10	3	0	0
both	78	17	5	0	0

Table 6.24: Manual Evaluation - model - Structure and Coherence

n-tgt-sents	Very Good	Good	Barely Acceptable	Poor	Very Poor
1	18	2	0	0	0
2	17	3	0	0	0
3	17	3	0	0	0
4	15	4	1	0	0
5	11	5	4	0	0

Table 6.25: Manual Evaluation - model - Structure and Coherence by n-tgt-sents

Scenario	Very Good	Good	Barely Acceptable	Poor	Very Poor
corpus	78	17	5	0	0
model	78	17	5	0	0

Table 6.26: Manual Evaluation - comparison (corpus/model) - Structure and Coherence

Rep. Seth Moulton is expected to announce Friday that he is ending his bid for the Democratic presidential nomination, ending his longshot candidacy for the White House. The three-term congressman and former Marine Corps soldier will speak at the Democratic National Committee's annual summer meeting in San Francisco. "Today, I want to use this opportunity, with all of you here, to announce that I am ending my campaign for president," he said in a video posted online, according to The Boston Globe. "I will say this: Today is the end of my campaign, and I know that we raised issues that are vitally important to the American people and our future," he added.

Figure 6.9: Example - model - WCEP - Structure and Coherence - Barely Acceptable

6.3 Discussion

On our evaluation we assess the quality of our generated training examples and the performance of the trained model. We examined four categories Omission, Inaccuracy Intrinsic, Inaccuracy Extrinsic and Structure and Coherence.

Corpus: After the analysis of the data we see that the training examples often omit information in the target that appears in the source sentences (54/100 - High Omission). An increase of the extraction threshold could descend the level of omission. We also observe that the information from the source compound, that appears in the target, is not changed - so the target preserves the meaning of the source.

Furthermore many times new information is added in the target, that does not appear in the source (46/100). Since the target-segment is part of a news article, the statements can be verified through the respective news article. The Structure and Coherence of the target segment is good in the majority of cases. We can conclude that the simple detection mechanism in our example generated pipeline works well.

Model: In case of the model output the omission rate of information that appears in the source and not in the target is lower than in the training examples (37/100 - High Omission). We can see that the training process has positive influence on the intake of information. We note a high level of wrongly transformed sentences in the target segment (46/100). This means that the learned text transformation often changes meaning of statements from the source in the target. In addition it also adds new information many times (44/100), that mostly can not be verified (30/100). The model adapts this property from the training data. The resulting output is a well structured and coherent text in the majority of cases (95/100). We see that our approach fits the extended sentence fusion task, nevertheless it struggles with typical issues of abstractive text generation.

Chapter 7

Conclusion

This work extends the task of sentence fusion from single output sentences to longer text segments. Previous work applies sentence fusion only on few input sentences and combines them to a single fusion sentence or a short paragraph. We examined a fusion mechanism that allows combining up to eleven input sentences to a cohesive text, which consists of multiple sentences. Instead of inserting single connectives in or between sentences our model performs an abstractive text transformation that reassembles the information from the source in the target text.

We follow up on a deep-learning approach and train a neural network on a task specific dataset. In order to do so, we build an extensive data-processing pipeline and exploit two text document corpora with it. To configure the pipeline we conduct several experiments to find proper metrics and threshold values.

After the processing of the dataset, we finetuned a state-of-the-art sequence-to-sequence model on our downstream task. Finally we evaluate the data-processing pipeline and the finetuned model. The manual evaluation gives us indications that further adjustment of the pipeline parameters could improve the quality of the training examples.

We conclude that our approach was successful and our model is capable performing the task. Nevertheless it struggles with typical issues of abstractive text transformation, it tends to change meaning and adds new information. In addition some information of the input sentences is not covered by the output text and therefore the compression ratio seems to high. Further optimization of the pipeline could find remedy.

The performance of the model depends on the context of application and further research is necessary to figure out the requirements of application use cases. The approach taken here can reassemble information in a complete different way and achieves good results regarding the structure and coherence of the resulting text. But other sentence fusion techniques, that just insert connectives and go without

such intensive text transformation are not confronted with issues named above. Though, they ignore noise in the input and process all content without further distinguishing of relevant, redundant or irrelevant information. Multiple techniques with different ad- and disadvantages exist, but it is not clear which method need to be applied in a certain application use case as long as the requirements are not consolidated.

As a result we have proved our approach as working and can point out advantages and disadvantages. We see further research as necessary to tackle the task of sentence fusion to cohesive text in a certain application use case.

Chapter 8

Appendix

Multi-News	WCEP
1. clean documents: removal delimiter symbols, etc. (like NEW_LINE) 2. remove duplicates of input articles ($editdistance(doc_0, doc_1) \leq 150$) 3. attend linguistic features; separate sentences 4. Clean sentences: removal of sentences with $number\_of\_tokens < 2$ and $number\_of\_characters < 10$ 5. manual annotation of relations between article and summary sentences	1. attend linguistic features; separate sentences (articles, summary) 2. filter instances with single sentence as summary, filter articles with less than 8 sentences 3. remove duplicates of input articles ($editdistance(doc_0, doc_1) \leq 150$) 4. rank articles by jaccard score with summary; take the top 5 input articles 5. Clean sentences: removal of sentences with $number\_of\_tokens < 2$ and $number\_of\_characters < 10$ 6. manual annotation of relations between article and summary sentences

Table 8.1: Extraction holdout-set - annotation procedure

Property / Corpus	Multi-News	WCEP
#instances	25	25
#mappings (all/not-empty) (not-empty: mappings without non-related sentences)	288 / 194	57 / 55
#relations (number of relations among all mappings)	381	394
max size of mapping (size: number of relations in mapping)	5	21
avg. number of relations all/not empty	1.32 / 1.96	6.91 / 7.16
distribution of mapping size (number of relations : number of corresponding mappings)	0:94, 1:79, 2:61 3:41, 4:8, 5:5	3:6, 2:6, 9:6, 8:6, 6:4, 11:4, 5:4, 4:4, 16:3, 1:3, 7:3, 0:2, 13:2, 14:1, 15:1, 21:1, 10:1

Table 8.2: Holdout-set statistics

metric	threshold	precision (Multi-News)	recall (Multi-News)	f-measure (Multi-News)	precision (WCEP)	recall (WCEP)	f-measure (WCEP)
sentence-embeddings	0.1	0.17	0.63	0.27	0.09	1.00	0.17
	0.2	0.17	0.63	0.27	0.10	1.00	0.18
	0.3	0.17	0.63	0.27	0.10	1.00	0.18
	0.4	0.17	0.63	0.27	0.10	0.99	0.18
	0.5	0.17	0.63	0.27	0.10	0.98	0.18
	0.6	0.17	0.63	0.27	0.11	0.98	0.19
	0.7	0.17	0.63	0.27	0.11	0.96	0.20
	0.8	0.18	0.63	0.29	0.15	0.91	0.26
	0.9	0.26	0.53	0.35	0.43	0.50	0.46
content-word	0.1	0.20	0.70	0.31	0.17	0.97	0.29
	0.2	0.26	0.66	0.37	0.33	0.79	0.47
	0.3	0.40	0.51	**0.45**	0.52	0.51	**0.51**
	0.4	0.61	0.33	0.43	0.69	0.32	0.44
	0.5	0.75	0.19	0.31	0.81	0.15	0.25
	0.6	0.80	0.13	0.22	1.00	0.05	0.10
	0.7	0.82	0.10	0.17	1.00	0.02	0.04
	0.8	0.80	0.07	0.13	1.00	0.00	0.01
	0.9	0.78	0.06	0.10	1.00	0.00	0.00
ROUGE-1-2-L	0.1	0.23	0.57	0.32	0.23	0.70	0.34
	0.2	0.37	0.36	0.37	0.48	0.37	0.42
	0.3	0.57	0.22	0.32	0.68	0.16	0.25
	0.4	0.76	0.14	0.24	0.72	0.07	0.13
	0.5	0.81	0.10	0.18	0.83	0.03	0.06
	0.6	0.82	0.08	0.15	0.88	0.01	0.02
	0.7	0.86	0.06	0.12	1.00	0.01	0.01
	0.8	0.86	0.05	0.09	1.00	0.00	0.01
	0.9	0.87	0.03	0.07	1.00	0.00	0.00

Table 8.3: Extraction experiment - results

Bibliography

[BM05] Regina Barzilay and Kathleen R. McKeown. "Sentence Fusion for Multidocument News Summarization". In: *Computational Linguistics* 31.3 (Sept. 2005), pp. 297–328. ISSN: 0891-2017. DOI: 10.1162/089120105774321091. eprint: https://direct.mit.edu/coli/article-pdf/31/3/297/1798201/089120105774321091.pdf. URL: https://doi.org/10.1162/089120105774321091.

[Bro+20] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].

[Che+20] Wei-Fan Chen et al. "Abstractive Snippet Generation". In: *Proceedings of The Web Conference 2020*. WWW '20. Taipei, Taiwan: Association for Computing Machinery, 2020, pp. 1309–1319. ISBN: 9781450370233. DOI: 10.1145/3366423.3380206. URL: https://doi.org/10.1145/3366423.3380206.

[Dev+19] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[Dia07] Brazdil Dias. "New Functions for Unsupervised Asymmetrical Paraphrase Detection". In: *Journal of Software* 2.4 (2007), pp. 12–23. DOI: http://www.jsoftware.us/vol2/jsw0204-02.pdf.

[DL15] Andrew M Dai and Quoc V Le. "Semi-supervised Sequence Learning". In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes et al. Vol. 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper/2015/file/7137debd45ae4d0ab9aa953017286b20-Paper.pdf.

[Fab+19] Alexander R. Fabbri et al. *Multi-News: a Large-Scale Multi-Document Summarization Dataset and Abstractive Hierarchical Model*. 2019. arXiv: 1906.01749 [cs.CL].

[Fua+19] Tanvir Fuad et al. “Neural Sentence Fusion for Diversity Driven Abstractive Multi-Document Summarization”. In: *Computer Speech Language* 58 (May 2019). DOI: 10.1016/j.csl.2019.04.006.

[Gev+19] Mor Geva et al. *DiscoFuse: A Large-Scale Dataset for Discourse-Based Sentence Fusion*. 2019. arXiv: 1902.10526 [cs.CL].

[Gha+20] Demian Gholipour Ghalandari et al. *A Large-Scale Multi-Document Summarization Dataset from the Wikipedia Current Events Portal*. 2020. arXiv: 2005.10070 [cs.CL].

[Her+15] Karl Moritz Hermann et al. “Teaching Machines to Read and Comprehend”. In: *arXiv e-prints*, arXiv:1506.03340 (June 2015), arXiv:1506.03340. arXiv: 1506.03340 [cs.CL].

[HH90] Michael A. K. Halliday and Ruqaiya Hasan. *Cohesion in English*. English. 10. impr. English language series. London u.a.: Longman, 1990. ISBN: 9780582550414. URL: https://katalog.ub.uni-leipzig.de/Record/0-21241934X.

[HR18a] Jeremy Howard and Sebastian Ruder. *Universal Language Model Fine-tuning for Text Classification*. 2018. arXiv: 1801.06146 [cs.CL].

[HR18b] Jeremy Howard and Sebastian Ruder. *Universal Language Model Fine-tuning for Text Classification*. 2018. arXiv: 1801.06146 [cs.CL].

[Hua+20] Dandan Huang et al. *What Have We Achieved on Text Summarization?* 2020. arXiv: 2010.04529 [cs.CL].

[Hug21] Huggingface. *fine-tuning scripts for summarization*. 2021. URL: https://github.com/huggingface/transformers/tree/master/examples/pytorch/summarization (visited on 05/23/2021).

[Leb+19] Logan Lebanoff, Kaiqiang Song, et al. *Scoring Sentence Singletons and Pairs for Abstractive Summarization*. 2019. arXiv: 1906.00077 [cs.CL].

[Leb+20] Logan Lebanoff, John Muchovej, et al. *Understanding Points of Correspondence between Sentences for Abstractive Summarization*. 2020. arXiv: 2006.05621 [cs.CL].

[Lew+19] Mike Lewis et al. *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. 2019. arXiv: 1910.13461 [cs.CL].

[LH02] Chin-Yew Lin and Eduard Hovy. “Manual and Automatic Evaluation of Summaries”. In: *Proceedings of the ACL-02 Workshop on Automatic Summarization - Volume 4*. AS ’02. Phildadelphia, Pennsylvania: Association for Computational Linguistics, 2002, pp. 45–51. DOI: 10.3115/1118162.1118168. URL: https://doi.org/10.3115/1118162.1118168.

[Li+18] Yutong Li et al. “Extraction Meets Abstraction: Ideal Answer Generation for Biomedical Questions”. In: *Proceedings of the 6th BioASQ Workshop A challenge on large-scale biomedical semantic indexing and question answering*. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 57–65. DOI: 10.18653/v1/W18-5307. URL: https://aclanthology.org/W18-5307.

[Liu+19] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL].

[LRU14] Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman. *Mining of massive datasets*. English. Second edition. Literaturangaben. Cambridge: Cambridge University Press, 2014. ISBN: 1107077230. URL: https://katalog.ub.uni-leipzig.de/Record/0-793858240.

[MK05] Erwin Marsi and Emiel Krahmer. “Explorations in Sentence Fusion”. In: *Proceedings of the Tenth European Workshop on Natural Language Generation (ENLG-05)*. Aberdeen, Scotland: Association for Computational Linguistics, Aug. 2005. URL: https://aclanthology.org/W05-1612.

[Nav01] Gonzalo Navarro. “A Guided Tour to Approximate String Matching”. In: *ACM Comput. Surv.* 33.1 (Mar. 2001), pp. 31–88. ISSN: 0360-0300. DOI: 10.1145/375360.375365. URL: https://doi.org/10.1145/375360.375365.

[NFC18] Mir Tafseer Nayeem, Tanvir Ahmed Fuad, and Yllias Chali. “Abstractive Unsupervised Multi-Document Summarization using Paraphrastic Sentence Fusion”. In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 1191–1204. URL: https://aclanthology.org/C18-1102.

[Pau20] Alejo Paullier. *MinHash LSH for document clustering 120 - Comment: Author apaullier commented on 25 Feb 2020*. 2020. URL: https://

github.com/ekzhu/datasketch/issues/120#issuecomment-590987788 (visited on 02/20/2020).

[Pet+18] Matthew E. Peters et al. *Deep contextualized word representations*. 2018. arXiv: 1802.05365 [cs.CL].

[Son+18] Yiping Song et al. *An Ensemble of Retrieval-Based and Generation-Based Human-Computer Conversation Systems*. 2018. URL: https://openreview.net/forum?id=Sk03Yi10Z.

[Vas+17] Ashish Vaswani et al. "Attention is All You Need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 9781510860964.

[Wik21a] Wikipedia. *Assignment problem — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Assignment%20problem&oldid=1043460561. [Online; accessed 30-September-2021]. 2021.

[Wik21b] Wikipedia. *Jaccard index — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Jaccard%20index&oldid=1040991564. [Online; accessed 03-October-2021]. 2021.

[Wik21c] Wikipedia. *Subsequence — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Subsequence&oldid=1042363172. [Online; accessed 30-September-2021]. 2021.

[Yan+19] Zhilin Yang et al. "XLNet: Generalized Autoregressive Pretraining for Language Understanding". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/file/dc6a7e655d7e5840e66733e9ee67cc69-Paper.pdf.

[Yas+19] Michihiro Yasunaga et al. "ScisummNet: A Large Annotated Corpus and Content-Impact Models for Scientific Paper Summarization with Citation Networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (July 2019), pp. 7386–7393. DOI: 10.1609/aaai.v33i01.33017386. URL: https://ojs.aaai.org/index.php/AAAI/article/view/4727.