SumTitles: a Summarization Dataset with Low Extractiveness

Valentin Malykh
Huawei Noah's Ark lab,
Moscow, Russia

Konstantin Chernis
Huawei Noah's Ark lab,
Moscow, Russia

Ekaterina Artemova
Huawei Noah's Ark lab,
HSE University
Moscow, Russia

Irina Piontkovskaya
Huawei Noah's Ark lab,
Moscow, Russia

Abstract

The existing dialogue summarization corpora are significantly extractive. We introduce a methodology for dataset extractiveness evaluation and present a new low-extractive corpus of movie dialogues for abstractive text summarization along with baseline evaluation. The corpus contains 153k dialogues and consists of three parts: 1) automatically aligned subtitles, 2) automatically aligned scenes from scripts, and 3) manually aligned scenes from scripts. We also present an alignment algorithm which we use to construct the corpus.[1]

1 Introduction

As most written communication is held in the form of dialogues, the amount of dialogue data increases over time. This poses a requirement for efficient dialogue mining tools for information extraction, search, and natural language understanding. An attractive path towards efficient information search is a compact representation, e.g., in the form of summarization. Although summarization methods could ease processing a large amount of textual data, few are applicable to dialogues. The main reason for this is that most of the datasets used for summarization models are completely different. Probably, the most well known corpus for summarization, CNN/DailyMail (Hermann et al., 2015), comprises news articles, while other cover social media posts (Völske et al., 2017a) and web documents (Chen et al., 2020). Deep summarization methods, both extractive (Zhong et al., 2020) and abstractive (Lewis et al., 2019), show high performance for these datasets. Dialogue data poses new challenges: first, a dialogue presents a conversation of two or more people, while news articles or social media posts are written by one person only. This means that multiple points of view can be expressed, and all of them need to be accounted for in the summary. Second, the grammar and the structure of the utterances differ drastically: more personal pronouns and colloquial expressions are used. Finally, the conventional sentence order is distorted: two consequent sentences may not be semantically related. These challenges limit the application of extractive summarization methods and push towards abstractive ones.

This paper is based upon an early work in dialogue dataset construction, namely AMI (Carletta et al., 2005). AMI corpus is small in size (141 dialogues) and does not allow the deep models training. In this paper, we try to overcome this significant flaw of AMI by developing a new large-scale dataset, a thousand times larger. We gather movies subtitles and freely available synopses into a dataset we call "corpus for *Sum*marization of movie sub*Titles*" (**SumTitles**). Section 3 describes SumTitles in details. The construction of the SumTitles dataset is a challenging task, and we present an algorithm to create alignment for scripts in Section 4. To compare SumTitles to the other summarization datasets, we introduce a new methodology, which is described in Section 5. The results of the comparison are also provided there. We experiment with state-of-the-art summarization approaches to create summaries for the dialogues (Liu and Lapata, 2019; Lewis et al., 2019) and therefore Section 6 contains the description of the baselines and the results achieved for SumTitles dataset and state-of-the-art results for CNN/DailyMail dataset for comparison. Section 7 conclude the paper and outline the future research directions.

[1] `https://github.com/huawei-noah/sumtitles`

The main **contributions** of this paper are the following. First, we present a novel dataset, SumTitles, which aims at dialog summarization (Section 3) and a splitting algorithm which we used to construct the corpus (Section 4). Secondly, we use multiple state-of-the-art summarization models on the SumTitles dataset to establish baselines (Section 6). Last but not least, we propose a novel Extractiveness Coefficient, based on which we conduct the comparison of existing datasets and SumTitles (Section 5).

2 Related Work

2.1 Summarization Methods

Many aspects in text summarization have been studied extensively since the first papers (Luhn, 1958). Research in machine learning methods for summarization dates back to early 2000. TextRank (Mihalcea and Tarau, 2004) is a simple and unsupervised yet efficient method for extractive summarization and keyphrase extraction. Supervised methods began to emerge towards the middle of 2010's when the first annotated corpora were developed, and neural machine translation (NMT) architectures were adopted for the task. While trainable methods for extractive methods confine to selecting and rearranging passages from the source text (Nallapati et al., 2017), abstractive methods involve generation plausible and fluent summaries from scratch. Sequence-to-sequence architectures, when conditioned of the source text and supervised for word prediction, are capable of composing a summary, though they suffer from several drawbacks. Their ability to handle unknown words and generate readable text seems to a certain extent limited. The first issue was alleviated by augmenting the standard sequence-to-sequence attentional model with a pointing network (See et al., 2017), which can copy words from the source text. To avoid generating redundant and repetitive summaries, a new training paradigm was proposed to combine the standard training objective is combined with reinforcement learning (Paulus et al., 2017). This helps to reduce exposure bias and improve the quality of generated summaries. Alternative training objectives include (Li et al., 2019) Determinantal Point Processes, producing better attention distributions in seq2seq models, improving thus both summary quality and diversity.

As of the late 2010s, transfer learning and transformer-derived language models are thoroughly integrated into the vast majority of natural language processing tasks. BertSumExt (Liu and Lapata, 2019) showcase how BERT can be usefully applied in extractive summarization by re-using of special token embeddings to represent and classify sentences. Abstractive summarization, in general, has seen a great deal of recent work. T5 (Raffel et al., 2019) is a unified sequence-to-sequence framework, which is pre-trained with a language model objective and fine-tuned for a number of downstream tasks, each treated as a text generation task. BART (Lewis et al., 2019), being a denoising autoencoder, is trained to reconstruct corrupted input. The reconstruction loss helps the model develop an efficient copying mechanism, which is core for abstractive models.

2.2 Datasets

Newspapers are a significant source for summarization data. Such low-scale datasets as **DUC 2002** (Over and Liggett, 2002) and bf TAC 2008 (Dang and Owczarzak,) comprise almost 600 and 1000 English news articles, correspondingly, aiming at single-document and multi-document summarization. **CNN/Daily Mail** (Hermann et al., 2015) is one of the most studied datasets, which being large enough, suits for both extractive and abstractive summarization. Gigaword (Rush et al., 2017) consists of news articles and corresponding headlines and can be treated as a source dataset for very short summaries. Social media can be seen as a more diverse source: WikiHow (Koupaee and Wang, 2018) and Webis-TLDR-17 (Völske et al., 2017b) comprise text and self-summaries written by different authors on a variety of subjects on WikiHow and Reddit platforms, correspondingly.

To the best of our knowledge, dialogue summarization has not received due attention so far. Two datasets, AMI (Carletta et al., 2005) and SAMSum (Gliwa et al., 2019), are the only datasets available for the task. AMI is a small dataset, created from meeting notes. It was re-designed in (Goo and Chen, 2018) to construct abstractive summarization dataset named DialSum. The initial 141 long dialogues were split to 7864 shorter ones. The topic descriptions are treated as summaries for these dialogues. Unfortunately, the speaker's information was lost in the transition. SAMSum was created by professional linguists, who were hired to, first, create chat-like dialogues, and second, annotate them with summaries. The SAMSum design focuses on information about the speakers and on the preservation of the messenger-like structure. The collected dialogues are considerably short, thus leading the summaries to be very extractive, as it is shown in Section 5. A similar issue was spotted by the authors of PersonaChat dataset (Zhang et al., 2018), where the dataset has been partially re-written after initial release due to often copied substrings from a person description to the utterances.

3 The SumTitles Dataset

Following (Gorinski and Lapata, 2015), we use movie scripts as the main source of data for corpus construction. The core concepts used for corpus construction are the following:

- The *subtitles* are captions for movies and series episodes. For our purpose *subtitles* are a joint text containing the utterances of the movie characters. The utterances separated by some special characters.
- A *script* is the text of movies and series episodes. Similarly to subtitles, it consists of utterances. However, each utterance is also labeled with the name of the movie character, whom this utterance belongs to. Typically, a script contains additional text, captioning a narrator's speech, which we do not use in our study.
- A *scene* is a subdivision of a movie or a series episode; the script consists of scenes. Each scene can be seen as a single dialogue. The scene is usually accompanied by a description of the internal or external space in which it occurs.
- A *plot summary* is a text summarizing a movie or a series episode contents in a few sentences.
- A *synopsis* is a text summarizing a movie or a series episode contents in the several paragraphs.
- A *cast* is a list of full character names, and sometimes their alter egos (i.e. "Tony Stark Iron Man").

The plot summaries, synopses, and casts are collected from the open sources, while for the subtitles and partially the scripts we use the existing datasets.

SumTitles consists of three parts: 1) Subtitles, 2) Scripts, and 3) Gold. Subtitles part has only rough alignment between the whole movie and a plot summary, since there is no information about characters and a scene separation. Scripts part comprises scenes which are automatically, but quite accurately, aligned with the synopses, most commonly a sentence per a scene. The last part, which we refer to as Gold, is labeled by human experts for an alignment between scenes and synopses.

The Subtitles part is an extraction from the OpenSubtitles corpus (2018 version), which is described in (Lison et al., 2019). We use only subtitles in English, and among them, only those which have plot summary available. We additionally filter the subtitles for the movies and series, which are not present in Scripts and Gold parts. The subtitles in OpenSubtitles dataset do not contain character names and scene separators. Thus we consider the whole subtitle to be a single dialogue of anonymous characters. The sample dialog accompanied with a movie plot is presented at Fig. 1. Although the subtitle could be split into several pieces to produce multiple dialogues, in this case, a plot summary will be covered by the split dialogue only partially.

The Scripts part itself consists of two parts: the movie scripts available from the open sources[2] and Friends series scripts described in (Chen and Choi, 2016). We consider the scripts for the movies which have synopses available only, while fortunately, Friends series has a synopsis for each episode[3]. We developed an algorithm, allowing us to split a synopsis in an automatic fashion to produce the dialogues accompanied with their summaries derived from the synopsis. The detailed description of the algorithm is available in Section 4.

[2]The scripts are collected from International Movie Scripts Database.

[3]We use plots published at Friends-TV.org: `http://www.friends-tv.org/epshort.html`

Plot

In the futuristic year of 2019, Los Angeles has become a dark and depressing metropolis, filled with urban decay. Rick Deckard, an ex-cop, is a "Blade Runner". Blade runners are people assigned to assassinate "replicants". The replicants are androids that look like real human beings. When four replicants commit a bloody mutiny on the Off World colony, Deckard is called out of retirement to track down the androids. As he tracks the replicants, eliminating them one by one, he soon comes across another replicant, Rachel, who evokes human emotion, despite the fact that she's a replicant herself. As Deckard closes in on the leader of the replicant group, his true hatred toward artificial intelligence makes him question his own identity in this future world, including what's human and what's not human.

Scene

- Care if I talk?
- I'm kind of nervous when I take tests.
- Just please don't move.
- Oh, sorry.
- I already had an IQ test this year.
- I don't think I've had one of these.
- Reaction time is a factor, so pay attention.
- Now answer as quickly as you can.
- Sure.
- 1187 at Hunterwasser.
- That's the hotel.
- What?
- Where I live.
- Nice place?
- Yeah, sure, I guess.
- Is that part of the test?
- No. Just warming you up. That's all.
- It's not fancy or anything.
- You're in a desert, walking along in the sand, when...
- Is this the test, now?
- Yes. You're in a desert, walking along in the sand...
- ...when you look down...
- What one?
- What?
- What desert?
- Doesn't make any difference.
- It's completely hypothetical.

Figure 1: An example of a movie plot accompanied with a subtitles excerpt from Subtitles part of SumTitles dataset. The original scene is an opening one from movie "Blade Runner" (1982).

Syn.

Ron goes to hospital again and Eve tries to help him because he is giving a hard time to nurse Frazin, but Ron is being jerk to her, shouting that he doesn't need a nurse but a doctor.

Scene

NURSE FRAZIN: Dr. Sevard's not on today.
RON: Do I look like I can wait til tomorrow?
NURSE FRAZIN: If you'll tell me what the problem is...
RON: Problem? Which problem you want to hear about? My lungs bleeding, my skin crawling, the jackhammer in my head... hell that's just the beginning of my problems sweetheart.
EVE: Mr. Woodroof?
RON: I don't want no nurse. I want a doctor. A goddamn doctor! Today! NOW!
EVE: Fine. How can I help you?
RON: Are you f***in' deaf, lady?
EVE: No. I'm a f***ing doctor!
EVE : If you want to discuss your list of problems, you can meet me in my office in twenty minutes.
RON: Twenty minutes?

Figure 2: An example of synopsis-scene pair from Scripts part of SumTitles dataset. The original scene comes from movie "Dallas Buyers Club" (2013).

The movies in the Gold part are picked from Scripts, but human experts controlled the splitting. The statistics of the dataset is available in Table 1. A sample from the Scripts part is presented in Figure 2. Interestingly, the sample alignment was achieved automatically.

4 Splitting a Script

A synopsis consists of the sentences, which we consider independent as each sentence describes a separate scene. To dampen the effects of this strong assumption, we develop an algorithm to split and join script scenes and sentences from a synopsis. The algorithm is presented as Algorithm 1.

Corpus \ Metric	#Samples	Ave. Speakers	Ave. Text Utt-s	Ave. Sum. Sents	Ave. Text Tokens	Ave. Sum. Tokens
News summarization						
CNN/DailyMail	312,084	N/A	N/A	3.75	781	56
Dialog summarization						
AMI	142	4.02	833.72	8.72	6041.5	178.65
DialSum	7864	N/A	9.00	1.00	72.54	3.66
SAMSum	16,369	2.09	9.13	1.92	121.61	21.92
SumTitles: Subtitles	131,864	N/A	852.65	4.12	6405.60	84.93
SumTitles: Scripts	21,469	4.88	28.44	3.75	423.06	55.03
SumTitles: Gold	290	5.52	26.77	3.57	394.80	51.02

Table 1: Summarization datasets' statistics.

Pre-processing is conducted in several steps. Firstly, we substitute scene speakers with cast character names, listed in the movie description. To this end, we estimate the similarity between scene speakers and character names by means of symbol-level n-gram Jaccard similarity. Next, we split each scene and each synopsis into separate sentences. Then we embed every sentence to get the vector representations. We use the pre-trained Universal Sentence Encoder model, described in (Cer et al., 2018)[4].

To implement the algorithm, we use several functions and formulae, which are referred to in a similar manner. We use Jaccard similarity to compare sentences and cosine similarity to compare vector representations. `Merge` function is merging the input of scenes list into one scene which collects all the utterances, annotated synopsis sentences, and character lists from the input scenes in the order of appearance. `LastSynId` and `FirstSynId` are returning the last and first (respectively) synopsis sentence indices from the ones annotated to an input scene. `Len` returns length of an input set, `Append` appends an input element to an input set. `Max`, `Mean`, `Union`, `Intersect`, `Sort`, and `Argmax` function according to their names.

There are additional helper functions presented as algorithms: `JaccardBest` (Alg. 3), `BestSplit` (Alg. 5), `Annot` (Alg. 4). Also there are two functions important for similarity computation: `CastSimilarity` (Alg. 6) and `TextSimilarity` (Alg. 7). The output of these two functions is a base for the splitting algorithm. Their description could be found in the appendix A.

Also the splitting algorithm is using `RestrictedDTW` presented as Algorithm 2. It is a modification of classic dynamic time-warping algorithm (Vintsyuk, 1968). In our case the restriction is that each cluster should contain exactly one synopsis sentence. If necessary, we add padding symbols to fill in the scenes that are speech free and do not have any utterances and actual sentences.

The splitting algorithm (Alg. 1) has several hyper-parameters: $\alpha, \beta, \gamma, \delta$, which are the weight coefficients for different similarity measures computed on the input data. These hyper-parameters are chosen based on the algorithm performance on the held out Gold part.

4.1 Splitting Quality

The hyper-parameters hyper-parameters $\alpha, \beta, \gamma, \delta$ could be tuned to achieve better splitting. We need to define a quality for a split. We use three measures to represent quality of proposed split.

The first measure is Accuracy, which is defined as following:

$$Accuracy = \frac{1}{N \cdot M} \sum_{i=1}^{N} \sum_{j=1}^{M} EQV(I_{ij}^{a}, I_{ij}^{h}), \quad (1)$$

where N is number of scenes, M is number of synopsis sentences, EQV is an equivalency function, i.e. its operands should be equal to each other, I_{ij} is an indicator function, whether scene i corresponds to sentence j, the indicator function could be ^{a} for algorithmic one, and ^{h} for human one.

[4]The model is available here: `https://tfhub.dev/google/universal-sentence-encoder-large`

Algorithm 1: Alignment algorithm for the scenes and sentences.

```
Data: $scenes$ - list of scenes, $syn\_sents$ list
      of synopsis sentences, $cast, \alpha, \beta, \gamma, \delta$
Result: alignment for the scene and sentences
$jac, osj \leftarrow$
  $CastSimilarity(scenes, syn\_sents, cast)$;

$sim\_max, sim\_mean \leftarrow$
  $TextSimilarity(scenes, syn\_sents)$;
$sim \leftarrow \alpha \cdot sim\_max + \beta \cdot sim\_mean$;
$sim \leftarrow sim + \gamma \cdot jac + \delta \cdot osj$;
$syn2scene \leftarrow RestrictedDTW(sim)$;
$scenes \leftarrow$
  $Annot(scenes, syn\_sents, syn2scene)$;
$ids \leftarrow Sort(syn2scene)$;
$m \leftarrow [Merge(scenes[: ids[0]])]$;
for $i \leftarrow 0$ to $Len(ids)$ do
    $c, n \leftarrow ids[i], ids[i+1]$;
    $bs \leftarrow BestSplit(scenes, c, n, sim)$;
    $m[-1] \leftarrow$
      $Merge([m[-1]] + scenes[c+1 : bs])$;
    $Append(m, Merge(scenes[bs : n]))$;
$m[-1] \leftarrow$
  $Merge([m[-1]] + scenes[ids[-1]+1 :])$;
```

Algorithm 2: `RestrictedDTW` algorithm.

```
Data: similarity matrix $S$ of size $m \times n$
Result: mapping for synopses to scenes
initialize $sim$ matrix of size $(2m+1, n)$ with
  zeros;
initialize $d$ matrix of size $(2m+1, n)$ with
  $-\infty$;
initialize $parent$ matrix of size $(2m+1, n)$
  with $-1$;
$sim[1 :: 2, :] \leftarrow S$;
$d[0, 0] \leftarrow 0$;
initialize $dir$ as an empty array;
for $i \leftarrow 1$ to $n$ do
    for $j \leftarrow 1$ to $m$ do
        $dir \leftarrow [d[i-1, j], d[i-1, j-1]]$;
        if $i \bmod 2 = 1$ then
            $Append(dir, d[i, j-1])$;
        $parent[i, j] \leftarrow Argmax(dir)$;
        $d[i, j] \leftarrow$
          $sim[i-1, j-1] + Max(dir)$;
$map \leftarrow [(-1, 0), (-1, -1), (0, -1)]$;
$syn \leftarrow 2m$;
$sc \leftarrow n - 1$;
initialize array $syn2scene$ of size $(2m+1)$
  with zeros;
while $syn \cdot sc \neq 0$ do
    $syn2scene[syn - 1] = sc - 1$;
    $change \leftarrow map[parent[syn, sc]]$;
    $syn \leftarrow syn + change[0]$;
    $sc \leftarrow sc + change[1]$;
return $syn2scene[1 :: 2]$;
```

Precision, the second measure, is defined as:

$$Precision = \frac{\sum_{i=1}^{N} \sum_{j=1}^{M} I_{ij}^{a} \cdot I_{i}^{h} j}{\sum_{i=1}^{N} \sum_{j=1}^{M} OR(I_{ij}^{a}, I_{ij}^{h})}, \tag{2}$$

where OR is a disjunction function, which returns 1 if at least one operand is 1.

And the last one is Recall. It is formulated as follows:

$$Recall = \frac{\sum_{i=1}^{N} \sum_{j=1}^{M} I_{ij}^{a} \cdot I_{ij}^{h}}{\sum_{i=1}^{N} \sum_{j=1}^{M} I_{ij}^{h}}, \tag{3}$$

We have randomly chosen two movie titles from Gold part of the dataset to tune the parameters onto (The Avengers, 2012 and 12 Monkeys). With this parameters we achieve 88.0% of accuracy, 27.0% of Precision, and 40.9% of Recall on the Scripts part.

5 Comparison to the Other Datasets

There are two aspects which could be considered for comparison: the size of the dataset and its extractiveness. The size of the dataset could be measured in different measurement units (number of speakers, tokens, utterances, summary sentences and summary tokens). The collected statistics for the datasets are presented in the Table 1. One could see that the number of samples, size of documents and summaries in our dataset is comparable to CNN/DailyMail, while the number of participants in our dataset is close to AMI corpus.

We introduce a compositional approach to define what extractiveness is. The existing approaches, such as (Grusky et al., 2018; Cibils et al., 2018), are evaluating different aspects of the extractivity itself such as coverage and fragment. Our goal is to capture the complex phenomenon. We use already existing approaches and extend them to achieve the final Extractiveness Coefficient (EC).

5.1 Extractive Score

As the first part to EC we use extractive score proposed in (Cibils et al., 2018). It is a metric measuring to what degree is a summary extracted from an input text. It accounts for long substrings of the source text, which occur in the summary. It is defined as follows:

$$ext.\ score(S) = \sum_{s \in P(ACS_s)} s \times (e^{s-1} - (1-s)/e), \tag{4}$$

where S is the summary, the ACS_s is the set of all long non-overlapping common sequences between S and the document, $P(ACS_s)$ is the set, where each element is the length of a common sequence divided by the length of the summary. This approach has a limitation of usage only the longest substrings, thus ignoring the short pieces which could be reused in the summary.

5.2 Extractive Oracle

The next part is so called "extractive oracle". This algorithm was proposed in (Liu and Lapata, 2019). It is a greedy algorithm aimed to generate an oracle summary for each document. The algorithm greedily select sentences from the input document which can maximize the ROUGE scores (Lin, 2004) against golden summary.

Essentially, the ROUGE metric is counting common token sequences in ground truth and system output sequences. There are three main variants: ROUGE-1, ROUGE-2, and ROUGE-L. ROUGE-1 and ROUGE-2 are using unigrams and bigrams respectively to compute a score. ROUGE-L is using longest common subsequence for a reference and a system output to compute the score. Here are the formulae for ROUGE metrics from original paper (Lin, 2004):

$$\text{ROUGE-N} = \frac{\sum_{r \in \{\text{references}\}} \sum_{w \in r} \text{Match}(w)}{\sum_{r \in \{\text{references}\}} \sum_{w \in r} \text{Count}(w)}, \tag{5}$$

where N stands for the length of a n-gram w, `Match` is the maximum number of n-grams co-occurring in a candidate summary (system output) and in a set of reference summaries, and `Count` is a number of all n-grams in references' set.

In particular, the ROUGE-N formulae mentioned above are describing how much the system output is capturing the reference summary and is often referred as the recall variant of ROUGE-N metrics, or simply RN-R or R$_L$-R for the "longest" variant. As there are no control over the length of the system output, so it can capture almost all of the reference summary while being excessively long. This issue is solved by the precision modification of ROUGE-N metrics that has the same formulae but the `Count` variable is now referred to the number of all n-grams in system output set. The ROUGE-N-F score is calculated as classical F_1 measure with ROUGE-N-Precision and ROUGE-N-Recall using harmonic mean (RN-F for short). We use F_1 variant of ROUGE-1, -2, & -L scores for this evaluation. This approach is free from the limitation of previous one and it evaluates both the long and the sort pieces. The limitation of this approach is immanent to its design: the existing phrases would capture only the main pieces from text in summary, while leaving aside pieces scatter around the text.

5.3 Summary-Input

The last part to EC is Recall-based ROUGE scores (uni-, bigram and longest) for the summaries interpreted as references against the input text used as system output. This approach is called to overcome the limitations of previous ones, it handles the scattered text pieces in the text. Although it has its own limitation - due to the nature of piece scattering, one could not measure the precision, only the recall of collected pieces.

5.4 BERT-Score

We also provide results of a metric aimed to capture the semantic similarity between the source text and its summary. We use a variant of BERT-Score defined in (Zhang et al., 2020) as following:

$$\texttt{BERT-Score} = \frac{\sum_{x_i \in x} idf(x_i)\ max_{y_j \in y}(x_i^T \cdot y_j)}{\sum_{x_i \in x} idf(x_i)}, \tag{6}$$

where idf is inversed document frequency, x is a set of token embeddings for a document text, and y is a set of token embeddings for a summary text.

Corpus \ Metric	EC	ext. score	extractive oracle			summary-input			BERT-Sc.
		(1e-4)	R1-F	R2-F	R$_L$-F	R1-R	R2-R	R$_L$-R	
News summarization									
CNN/DailyMail	77.31	175.56	54.09	32.35	50.51	90.12	51.25	87.32	90.94
Dialog summarization									
AMI	34.91	0.28	24.93	5.12	22.26	82.27	28.69	80.80	82.30
DialSum	8.46	2.35	9.53	0.51	9.02	18.72	1.06	18.06	32.90
SAMSum	95.52	384.58	51.38	22.63	49.05	68.95	25.59	66.44	83.48
ST: Subtitles	32.92	30.76	9.71	2.48	8.69	79.34	21.72	77.72	80.04
ST: Scripts	14.32	17.84	26.26	4.38	23.20	13.75	2.07	12.75	60.06
ST: Gold	16.70	10.31	26.19	3.86	23.26	26.19	3.86	23.26	58.28

Table 2: Extractiveness Coefficient (EC) and the other metrics for the considered datasets. ST stands for SumTitles.

5.5 Extractiveness Coefficient

Thus we decided to combine the previously described approaches (namely, extractive score, extractive oracle, and summary-input) and achieve the reasonable extractiveness evaluation. To compute the desired Extractiveness Coefficient we scale all scores so they are put in the same domain: ROUGE scores are multiplied by 100, and the extraction score is multiplied by 10000. Afterwards all the collected metrics are averaged.

The Table 2 contains the computed scores for several datasets. One could see that the collected dataset is much closer both to AMI corpus and to its variant DialSum than any previously presented one. BERT-Score measures the similarity of a text and its summary, basing on vector representations. One could mention that CNN/DailyMail and SAMSum datasets have high similarity, but also high extractiveness, while AMI has low extractiveness. Interestingly, DialSum dataset, composed from AMI using sliding window has significantly lower extractive score, but also BERT-Score one. As we hope the presented dataset passed between Scylla and Charybdis and while keeping low extractive score has comparatively high BERT-Score.

6 Experiments

To better understand our dataset's properties, we evaluate several current summarization models on it, accounting for both extractive and abstractive approaches.

We evaluate the baseline model in multiple settings:

- **no speakers** setting. In this setting, we used an anonymized version of SumTitles. To anonymize the dataset, we remove cast character names from the synopsis.
- **with speakers** setting. A non-anonymized version of SumTitles consists of concatenated cast character names and their utterances.

6.1 BertSumExt

BertSumExt model, introduced in (Liu and Lapata, 2019), treats extractive summarization as a binary sequence classification task to determine whether each sentence should be included in the summary. It utilizes BERT (Devlin et al., 2018) as an encoder and stacks several Transformer layers (Vaswani et al., 2017) on top of it with final softmax function to produce the logits. We use the speaker feature for BertSumExt training, which is an extension of an original utterance with cast character name.

6.2 BART

BART model, described in (Lewis et al., 2019), presents a denoising autoencoder pre-training objective (text masking and sentence shuffling), leveraged to improve model generation capabilities of the original Transformer architecture.

We evaluated the BART model with two additional training features. Firstly, we introduced special separator tokens, which was not used during the original BART pre-training procedure. The separator tokens are used to join the utterances. Secondly, we use speaker feature analogously to BertSumExt baseline.

6.3 Results

In this section we present the results for baseline algorithms on the collected SumTitles dataset. We explore several different ways of feeding utterances into the model, namely:

- concatenating all the utterances (default);
- representing each utterance in "Speaker: Utterance" format (capitalized speaker name separated with colon) and then concatenating (*w/ speakers*);
- adding separators between utterances (*w/ seps*): $[START]\ Utt_1\ [END]\ [START]\ Utt_2\ [END]$
- combined approach (*w/ speakers & seps*): $[START]\ Speaker_1\ [SEP]\ Utt_1\ [END]$.

We propose the following usage of the SumTitles: Subtitles part could be used for pre-training. Scripts part is used for training, and Annotated part is used for evaluation. As metrics we are using F_1 variant of ROUGE-1, -2, & -L. In our experiments, we truncated longer dialogues to 1024 tokens.

For the technical details, BertSumExt usage is used almost identical to the CNN/DailyMail experiments. The only change made is the number of generated sentences, which is set to 6, based on the train set statistics. This should account for shorter sentences. We use epoch checkpointing instead of steps due to a smaller dataset. As for the BART, we follow an original experiment design.

The Table 3 presents the evaluation results. The BART model, although showing higher results than BertSumExt, still demonstrates twice as low results in comparison with the results on CNN/DailyMail dataset (see Table 4). This relation keeps roughly the same for the extractive oracle results on the SumTitles dataset.[5]

7 Conclusion

We target creating a dataset, which will show the limitations of previously presented summarization datasets, which seem to borrow a lot from the original texts. We presented SumTitles dataset, which on the one hand, is significantly larger than the previous low extractive AMI/DialSum datasets. On the other hand, SumTitles is comparable in size with recent abstractive datasets, such as CNN/DailyMail, which are highly extractive. To compare the summarization datasets, we presented a methodology for extractiveness evaluation. The alignment of scripts and summaries proved to be a challenging task that we could solve with a specialized algorithm. This algorithm could be used to extend the current work and be adopted to other long texts to produce a split in semantically coherent units to facilitate training.

[5]The code for extractive oracle was released with BertSumExt model and is available online `https://github.com/nlpyang/BertSum`, although we have inspected the code and found that it has some issues. After fixing them, we have re-evaluated the results on CNN/DailyMail and achieved better performance than in original paper (Zhong et al., 2020). We report the improved results.

Model	R1	R2	R_L
Ext. oracle w/ speakers	26.19	3.86	23.26
Extractive			
BertSumExt w/ speakers	17.93	1.65	15.98
Abstractive			
BART	19.08	2.36	15.47
BART w/ seps	19.51	2.49	16.10
BART w/ speakers	20.77	2.71	16.75
BART w/ speakers & seps	21.16	3.10	17.36

Table 3: Evaluation of baseline runs on SumTitles by ROUGE F_1.

Model	R1	R2	R_L
Extractive oracle	54.09	32.35	50.51
Extractive			
BertSumExt	42.73	20.13	39.20
Abstractive			
BART	44.16	21.28	40.90

Table 4: Evaluation of baseline runs on CNN/DailyMail by ROUGE F_1.

There are a few directions for the future works. Firstly, the additional markup could be done to extend the Annotated part of the dataset. Secondly, major modifications to the current state of the art models are demanded to improve the performance on the dialogue summarization task. Thirdly, the proposed algorithm could be applied to other domains, such as fiction books.

Acknowldgements

Ekaterina Artemova is partially supported by the framework of the HSE University Basic Research Program and funded by the Russian Academic Excellence Project “5-100”.

References

Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Mael Guillemot, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, et al. 2005. The ami meeting corpus: A pre-announcement. In *International workshop on machine learning for multimodal interaction*, pages 28–39. Springer.

Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Lyn Untalan Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, Yun hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder. In *EMNLP demonstration*, Brussels, Belgium.

Yu-Hsin Chen and Jinho D Choi. 2016. Character identification on multiparty conversation: Identifying mentions of characters in tv shows. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 90–100.

Wei-Fan Chen, Shahbaz Syed, Benno Stein, Matthias Hagen, and Martin Potthast. 2020. Abstractive Snippet Generation. In Yennung Huang, Irwin King, Tie-Yan Liu, and Maarten van Steen, editors, *Web Conference (WWW 2020)*, pages 1309–1319. ACM, April.

André Cibils, Claudiu Musat, Andreea Hossman, and Michael Baeriswyl. 2018. Diverse beam search for increased novelty in abstractive summarization.

Hoa Trang Dang and Karolina Owczarzak. Overview of the tac 2008 update summarization task.

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

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 70–79.

Chih-Wen Goo and Yun-Nung Chen. 2018. Abstractive dialogue summarization with sentence-gated modeling optimized by dialogue acts. In *Proceedings of 7th IEEE Workshop on Spoken Language Technology*.

Philip Gorinski and Mirella Lapata. 2015. Movie script summarization as graph-based scene extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1066–1076.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 708–719.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *CoRR*, abs/1506.03340.

Mahnaz Koupaee and William Yang Wang. 2018. Wikihow: A large scale text summarization dataset. *arXiv preprint arXiv:1810.09305*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Lei Li, Wei Liu, Marina Litvak, Natalia Vanetik, and Zuying Huang. 2019. In conclusion not repetition: Comprehensive abstractive summarization with diversified attention based on determinantal point processes. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 822–832.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July. Association for Computational Linguistics.

Pierre Lison, Jörg Tiedemann, Milen Kouylekov, et al. 2019. Open subtitles 2018: Statistical rescoring of sentence alignments in large, noisy parallel corpora. In *LREC 2018, Eleventh International Conference on Language Resources and Evaluation*. European Language Resources Association (ELRA).

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3721–3731.

Hans Peter Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain, July. Association for Computational Linguistics.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Paul Over and Walter Liggett. 2002. Introduction to duc: an intrinsic evaluation of generic news text summarization systems. *Proc. DUC. http://wwwnlpir. nist. gov/projects/duc/guidelines/2002. html*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Alexander M Rush, SEAS Harvard, Sumit Chopra, and Jason Weston. 2017. A neural attention model for sentence summarization. In *ACLWeb. Proceedings of the 2015 conference on empirical methods in natural language processing*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada, July. Association for Computational Linguistics.

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

T. K. Vintsyuk. 1968. Speech discrimination by dynamic programming. In *Cybernetical System Analysis*, volume 4, pages 54–47.

Michael Völske, Martin Potthast, Shahbaz Syed, and Benno Stein. 2017a. Tl; dr: Mining reddit to learn automatic summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 59–63.

Michael Völske, Martin Potthast, Shahbaz Syed, and Benno Stein. 2017b. TL;DR: Mining Reddit to learn automatic summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 59–63, Copenhagen, Denmark, September. Association for Computational Linguistics.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *Proceedings of International Conference on Learning Representations*.

Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2020. Extractive summarization as text matching. *arXiv preprint arXiv:2004.08795*.

A Helper Functions

Algorithm 3: `JaccardBest` function.

```
Data: query, set of options
Result: closest to $query$ by Jaccard distance element from $options$
for $i \leftarrow 0$ to $Len(options)$ do
    $d[i] \leftarrow Jaccard(query, options[i])$;
return $options[Argmax(d)]$;
```

Algorithm 4: `Annot` function.

```
Data: $syn\_sents$ - set of synopsis sentenses, $scenes$ - set of scenes, $syn2scene$ - mapping of
      synopsis sentences to scenes
Result: $scenes$ annotated with $syn\_sents$ according to $syn2scene$ map
for $i \leftarrow 0$ to $Len(syn2scene)$ do
    $sc \leftarrow scenes[syn2scene[i]]$;
    $AddSynToScene(syn\_sents[i], sc)$;
return $scenes$;
```

Algorithm 5: `BestSplit` function.

```
Data: $scenes$ - list of scenes, $sc1$ - first scene index, $sc2$ - second scene index, $sim$ - similarity
      matrix
Result: best partition of not annotated scenes according to similarity matrix $sim$
$syn1 \leftarrow LastSynId(scenes[sc1])$;
$syn2 \leftarrow FirstSynId(scenes[sc2])$;
for $split \leftarrow sc$ to $sc2$ do
    $s1 \leftarrow Sum(sim[syn1, sc1 : split])$;
    $s2 \leftarrow Sum(sim[syn2, split : sc2])$;
    $score[split] \leftarrow s1 + s2$;
return $Argmax(score)$;
```

Algorithm 6: `CastSimilarity` function.

```
Data: $scenes$ - list of scenes, $syn\_sents$ list of synopsis sentences, $cast$
Result: similarity matrices of scene and synopsis characters
initialize $scene\_roles$ as array of size $Len(scenes)$ with empty arrays; for $i \leftarrow 0$ to $Len(scenes)$
 do
    for $speaker \in scenes[i]$ do
        $r \leftarrow JaccardBest(speaker, cast)$;
        $Append(scene\_roles[i], r)$;
initialize $syn\_roles$ as array of size $Len(syn\_sents)$ with empty arrays; for $i \leftarrow 0$ to
 $Len(syn\_sents)$ do
    for every window $w$ for $syn\_sents[i]$ do
        if there is a capitalized word in $w$ then
            $r \leftarrow JaccardBest(w, cast)$;
            $Append(syn\_roles[i], r)$;
initialize $jac$ and $osj$ with zeros;
for $i \leftarrow 0$ to $Len(syn\_roles)$ do
    $syn\_r \leftarrow syn\_roles[i]$;
    for $j \leftarrow 0$ to $Len(scene\_roles)$ do
        $sc\_r \leftarrow scene\_roles[j]$;
        $is \leftarrow Intersect(syn\_r, sc\_r)$;
        $un \leftarrow Union(syn\_r, sc\_r)$;
        $jac[i, j] \leftarrow Len(is)/Len(un)$;
        $osj[i, j] \leftarrow Len(is)/Len(syn\_r)$;
return $jac, osj$;
```

Algorithm 7: `TextSimilariry` function.

```
Data: $scenes$ - list of lists of scene sentences, $syn\_sents$ - list of synopsis sentences
Result: similarity matrices for scenes and synopsis sentences
for $i \leftarrow 0$ to $Len(syn\_sents)$ do
    $syn \leftarrow syn\_sents[i]$;
    for $j \leftarrow 0$ to $Len(scenes)$ do
        for $k \leftarrow 0$ to $Len(scenes[j])$ do
            $sim[k] \leftarrow CosineSim(syn, scenes[j][k])$;
        $sim\_max[i, j] \leftarrow Max(sim)$;
        $sim\_mean[i, j] \leftarrow Mean(sim)$
return $sim\_max, sim\_mean$
```