Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 6718–6724
Marseille, 11–16 May 2020

Align then Summarize: Automatic Alignment Methods for Summarization Corpus Creation

Paul Tardy[1, 2] **David Janiszek** [1, 3] **Yannick Estève** [4] **Vincent Nguyen** [2]
[1] LIUM – Le Mans Université
[2] Ubiqus Labs
[3] Université Paris Descartes
[4] LIA – Avignon Université
pltrdy@gmail.com, david.janiszek@parisdescartes.fr,
yannick.esteve@univ-avignon.fr, vnguyen@ubiqus.com

Abstract

Summarizing texts is not a straightforward task. Before even considering text summarization, one should determine what kind of summary is expected. How much should the information be compressed? Is it relevant to reformulate or should the summary stick to the original phrasing? State-of-the-art on automatic text summarization mostly revolves around news articles. We suggest that considering a wider variety of tasks would lead to an improvement in the field, in terms of generalization and robustness. We explore meeting summarization: generating reports from automatic transcriptions. Our work consists in segmenting and aligning transcriptions with respect to reports, to get a suitable dataset for neural summarization. Using a bootstrapping approach, we provide pre-alignments that are corrected by human annotators, making a validation set against which we evaluate automatic models. This consistently reduces annotators' efforts by providing iteratively better pre-alignment and maximizes the corpus size by using annotations from our automatic alignment models. Evaluation is conducted on `public_meetings`, a novel corpus of aligned public meetings. We report automatic alignment and summarization performances on this corpus and show that automatic alignment is relevant for data annotation since it leads to large improvement of almost +4 on all ROUGE scores on the summarization task.
Keywords: Alignment, Summarization, Corpus Annotation

1. Introduction

Automatic Text Summarization is the task of producing a short text that captures the most salient points of a longer one. However, a large variety of tasks could fit this definition.

Many factors are critical in the summarization process, such as whether to rephrase the source (abstractiveness) or use part of the source as-is (extractiveness); the length ratio of target and source (compression factor); the source and target lengths and their variances; and the information distribution – i.e. how important information is distributed along the text.

Most of summarization benchmarks (See et al., 2017; Paulus et al., 2017; Gehrmann et al., 2018) rely on news articles from CNN and DailyMail (Hermann et al., 2015; Nallapati et al., 2016) which exhibit particular characteristics such as: (i) being quite extractive i.e. picking portions of text from the source, the opposite of abstractive (Liu et al., 2018); (ii) a high compression factor with the summary being up to 10 times shorter than the source (Liu et al., 2018); (iii) a low variance in both source and target length, and (iv) concentrating information mostly at the beginning of the article: for example, papers working on the CNN-DailyMail corpus (Hermann et al., 2015; Nallapati et al., 2016) often truncate the article to the first 400 words of the article (See et al., 2017; Gehrmann et al., 2018; Ziegler et al., 2019), ignoring up to half of it.

In contrast, we explore meeting data, using transcription as the source, and the meeting report as the target.

Contrary to news articles, there is high variance in the length of speaker interventions; the data need to be rephrased into a written form (thus, an abstractive process by nature), and to be informative throughout. In this work, we focus on so-called *exhaustive reports*, which are meant to capture all the information and keep track of speaker interventions. Information itself is not summarized, but the speech is compressed from an oral form to a written one. Thus, the compression factor is lower than in news tasks but variance remains high, depending on how verbose the intervention is.

The data at hand consist of (i) exhaustive reports produced by Ubiqus in-house editors, (ii) full audio recording of the meeting. An automated transcript is produced from the latter with an automatic speech recognition system very close to the one described (Hernandez et al., 2018; Meignier and Merlin, 2010) but trained for French language from internal data.

Such data are not suitable for summarization learning as-is, therefore we propose to segment it at the intervention level (*i.e.* what is said from one speaker until another one starts). It is particularly convenient since the nature of our dataset ensures that all interventions remain (apart from very short ones) and chronological order is preserved in both the transcription and the report. Reports explicitly mention speakers, making segmentation trivial and error-free for that part. Transcriptions do not have such features so we present an alignment process that maps interventions from the reports with its related transcription sentences based on similarity.

We bootstrap the corpus creation, iterating between automatic pre-alignment generations and corrections from human annotators. We aim at jointly minimizing human effort while fine-tuning automatic alignment models to eventually use alignment models for automatic data annotation.

In this paper, we present a methodology for building a sum-

marization corpus based on the segmentation and alignment of reports and transcriptions from meetings using a bootstrapping approach. We also present `public_meetings`, a novel public meeting dataset, against which we evaluate both automatic alignment and summarization. Summarization models are first trained on the gold set – from human annotator –, and then using automatic annotations with our automatic alignment models which outperform the baseline by a large margin (almost +4 on all considered ROUGE metrics). Source code, data and reproduction instructions can be found at:
`https://github.com/pltrdy/autoalign`.

2. Related Work

This work aims to jointly segment two related files – a transcription and a report of the same meeting – so that the i-th segment of the report actually corresponds to the j-th segment of the transcription.

Since report side segmentation is simple thanks to its structure, we focus on the transcription side. Bearing that in mind, the task is similar to a linear segmentation problem, i.e. finding borders between segments. (Hearst, 1997) proposed TEXTTILING, a linear segmentation algorithm that compares adjacent blocks of text in order to find subtopic shifts (borders between segments) using a moving window over the text and identifying borders by thresholding. C99, as proposed by (Choi, 2000), uses similarity and ranking matrices instead, then clustering to locate topic boundaries. TEXTTILING has been extended (i) to audio signals (Banerjee and Rudnicky, 2006) but is said to lack robustness to atypical participant behavior (which is common in our context); (ii) to work with word embeddings in order to capture similarity between query and answer in a dialogue context (Song et al., 2016). (Alemi and Ginsparg, 2015) also explore word embedding use in segmentation by incorporating it into existing algorithms and showing improvements. (Badjatiya et al., 2018) address the segmentation task with an end-to-end attention-based neural approach. While such an approach could be investigated in the future, we could not consider it in this work due to the lack of reference data. (Glavas et al., 2016) use semantic relatedness graph representation of text then derive semantically coherent segments from maximal cliques of the graph. One issue of this approach is that searching for large segments in big texts requires decreasing the threshold which exponentially increases computational cost, eventually making our task intractable.

2.1. Alignment

Alignment has already been studied for corpus creation. In particular, (Barzilay and Elhadad, 2003; Nelken and Shieber, 2006) extract related segments from the *Encyclopedia Britannica* and *Britannica Elementary* (a simpler version). It is different from our work since we are looking for a total alignment, i.e. both documents must be fully aligned, not just partially extracted.

Furthermore, alignment of oral speech to its written form has been studied by (Braunschweiler et al., 2010) in the context of audio books and by (Lecouteux et al., 2012) for subtitles and transcripts (e.g. of news report) in order to improve Automatic Speech Recognition engines. While such approaches sound similar to ours, they mostly look for exact matches rather than an approximate alignment of asymmetrical data, based on textual similarity.

2.2. Summarization Datasets

(Hermann et al., 2015; Nallapati et al., 2016) proposed the first multi-sentence summarization dataset, with more than 280.000 training pairs. Sources are up to 800 words long (but are often truncated to the first 400 words (See et al., 2017; Gehrmann et al., 2018; Ziegler et al., 2019)) and the target is around 50 words on average. A similar dataset based on NY Times articles was presented by (Paulus et al., 2017), with three times more training pairs, sources of 800 words and targets of 45 words on average. (Liu et al., 2018) work on generating Wikipedia introductions (known as leads) from reference articles and web-crawled data. Both inputs and outputs are several orders of magnitude longer: sources can be up to 10^6 words and targets are in the $10^1 - 10^3$ range.

In our context, we are dealing with limited resources, in particular with respect to ready to train data – which motivated this paper. Our dataset comprises 20,000 gold standard training pairs, and up to 60,000 pairs when taking into account all the automatically aligned data. We currently filter training pairs in order to contain fewer than 1000 words and 50 sentences. Future work would explore a wider range of segment lengths.

3. Methods

Our task consists in finding the best alignment between a meeting transcription $\mathcal{T} = \{t_1, \dots, t_I\}$ and the related human written report $\mathcal{R} = \{r_1, \dots, r_J\}$.

Both documents are *segmented* into mutually exclusive sets of sentences $\hat{\mathcal{T}} = \{T_1, \dots, T_M\}$, $\hat{\mathcal{R}} = \{R_1, \dots, R_N\}$.

Alignment maps each transcription segment $T_m \in \hat{\mathcal{T}}$ to exactly one report segment $R_n \in \hat{\mathcal{R}}$ based on sentence-level similarities $S_{i,j} = score(t_i, r_j)$, with $t_i \in \mathcal{T}$, $r_j \in \mathcal{R}$.

The alignment process is a pipeline of different modules. The first one reads the data; the second one independently segments each side – respectively report and transcription; the third one computes similarity scores in order to find the alignment that maximizes the overall score. This section presents those modules.

3.1. Segmentation

Segmentation consists in finding borders in texts such that each segment can be processed independently. The segmentation granularity should be fine enough for segments to be not too long (which would make learning more difficult, and result in fewer training pairs) and coarse enough to remain relevant (very short segments cannot be meaningfully summarized). We consider speaker interventions – i.e. uninterrupted speech of only one speaker – to be an appropriate segmentation level. In particular, we make the assumption that the task of writing a report can roughly be divided into sub-tasks consisting of reports for each intervention, which is a close approximation of *exhaustive reports*.

On the report side, each speaker's intervention may be explicitly mentioned using special tags in the document (one particular style applied to names), or identified with rule-based identification e.g. looking for 'Mr.', 'Ms.' etc.
On the transcription side, segments are groups of sentences defined by the automatic speech recognition system.

3.2. Text Representation and Similarity function

The alignment process consists in finding, for each transcription segments T_m, its related report segment R_n, in other words the function:

$$alignment(m) = n, \forall m \in [1, M],\ n \in [1, N]$$
$$alignment(m) \leq alignment(m+1)$$

We consider a sentence-level similarity matrix S between the transcription $\mathcal{T}$ and the report $\mathcal{R}$ such as $S_{i,j} = score(t_i, r_j)$ with $(t_i, r_j) \in \mathcal{T} \times \mathcal{R}$.
For the *score* function, we experimented with (i) *ROUGE* from (Lin, 2004); (ii) cosine similarity on $tf \cdot idf$ representations ; (iii) cosine similarity based on word embedding vectors. A $pooling$ function (typically a sum) is applied to word embeddings to produce sentence embeddings, as shown in figure 1.
By default, both $\mathcal{T}$ and $\mathcal{R}$ are sets of sentences[1], however we also use sliding windows with overlap over sentences. For each document $D \in \{\mathcal{T}, \mathcal{R}\}$, the k-th sliding window $W^D_{o,s}$ is a set of s sentences having its first (respectively last) o sentences in common with previous window (resp. next).

$$W^D_{o,s}(k) = \{s_{ks-ko}, ..., s_{(k+1)s-ko} \mid s_i \in D\} \quad (1)$$

Sliding windows aggregate sentence representations into a single vector using the agg function (see figure 1, then we calculate scores for all pairs of sliding windows from both sides:

$$S^{sliding}_{k,l} = score\big(agg(W^T_{o,s}(k)), agg(W^R_{o,s}(l))\big) \quad (2)$$

then similarities are assigned back to the sentence level:

$$S_{i,j} = red\left(\left\{S^{sliding}_{k,l} \mid (s_i, s_j) \in \left(W^T_{o,s}(k) \times W^R_{o,s}(l)\right)\right\}\right) \quad (3)$$

The reduction function red (sum or product) calculates sentence scores from the sliding windows that contain it.

3.3. Alignment

Having sentence level (of sentence-windows) similarities of every pairs of transcription and report, the alignment task is now to maximize the similarity at the document level. We use dynamic programming, which aims to find an optimal path in the similarity matrix while ensuring – by design – that transcription and report are aligned chronologically.
We introduce the *alignment matrix* A that, for each coordinate (i, j) corresponds to the similarity (eventually scaled to the power of p) plus the maximal value from its top $(i, j-1)$ or left $(i-1, j)$ neighbor coordinates :

$$A_{i,j} = {S_{i,j}}^p + \max(A_{i-1,j}, A_{i,j-1})$$
$$A_{1,1} = {S_{1,1}}^p \quad (4)$$

[1]Sentences are determined by punctuation, which is predicted by the speech recognition system on the transcription side.

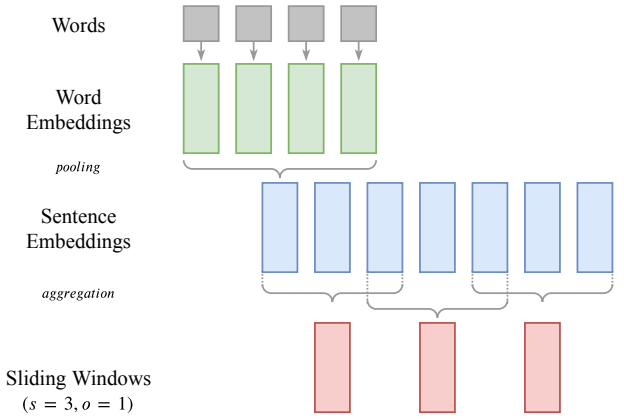

Figure 1: Text Representations: from words to sliding-windows.

At each position (i, j) we keep track of the previous position (e.g. either $(i-1, j)$ or $(i, j-1)$):

$$H_{i,j} = \underset{c \in \{(i-1,j);(i,j-1)\}}{\arg\max} (A_c) \quad (5)$$

ultimately giving us the optimal path, which correspond to the sentence level alignment P:

$$P_{k-1} = H_{i,j} \text{ with } (i, j) = P_k$$
$$P_{I+J} = (I, J)$$

Figure 2 shows simple example of the alignment process.
To derive the segment-level alignment of a transcription segment T_i we choose R_j to maximize similarity along the path:

$$alignment(m) = \underset{n}{\arg\max}\left(\sum_{s_i \in T_m} \sum_{s_j \in R_n} A_{i,j} \mathbb{1}_{(i,j) \in P}\right) \quad (6)$$

3.4. Evaluation

Linear segmentation performance is measured using `WindowDiff` (Pevzner and Hearst, 2002), which compares boundaries predicted by the algorithm to the reference in a moving window of size k. `WindowDiff` is based on P_k (Beeferman et al., 1999) but is meant to be fairer, with respect to false negatives, number of boundaries, segment size and near miss errors. We report `WindowDiff` scores for our experiments. We also consider simple metrics such as the segment accuracy and word accuracy. Experience scores are micro-averaged over reference files.

4. Experiments

4.1. Bootstrapping the corpus creation

To build a corpus from scratch we iterate over three phases, (i) generating pre-alignments from the data using an automatic alignment model; (ii) correct the pre-alignment thanks to human annotators to get a gold reference set; (iii) evaluate models with respect to the new reference set.
Iterations increase the amount of gold references, allowing accurate evaluation of automatic alignment models, eventually making the annotators' task easier.

	t_1	t_2	t_3	t_4
r_1	5	3	8	9
r_2	5	7	6	2
r_3	3	4	7	5

	t_1	t_2	t_3	t_4
r_1	5 →	8 →	16 →	25
	↓			↓
r_2	10 →	17 →	23	27
	↓	↓	↓	
r_3	13	21	30 →	35

Figure 2: Example of Dynamic Programming algorithm finding optimal path. At each coordinates (i, j), the alignment (A, on the right) adds the corresponding similarity from S (on the left) to highest neighbor value, either from top $(i, j-1)$ or left $(i-1, j)$, as shown with arrows (equivalent to H); red arrows represent the optimal path P. Similarity values are arbitrary here for simplicity.

Gold Alignments We developed an ad-hoc platform to collect gold alignments thanks to human annotators to serve as reference sets. We use our automatic alignment models to provide a pre-alignment that is then corrected by the annotator.

Grid Search In order to evaluate a wide variety of parameters at a reasonable computational cost, we use several validation sets varying in their amount of reference files. The evaluation process iteratively selects best parameters, thus reducing their number, then evaluates these sub-sets on a bigger reference set. It helps us to efficiently explore the parameter space without spending too much effort on obviously sub-performing parameter sets and eventually identify most critical parameters.

1rst Iteration: diagonal alignment The first iteration started without any reference file, therefore, we had no way of quantitatively evaluating the auto-alignment process. Still, in order to provide a pre-alignment to human annotator, we used a naive approach that aligns segments diagonally: we do not compute similarity ($S_{i,j} = 1, \forall(i, j)$) and move into the alignment matrix to stay on a diagonal i.e. we replace the position history matrix H of eq. 5 to be:

$$H_{i,j} = \begin{cases} (i-1, j) \text{ if } r_{i,j} < r \\ (i, j-1) \text{ otherwise} \end{cases}$$
$$\text{with } r = |\,T\,| \,/\, |\,R\,|$$
$$\text{and } r_{i,j} = (i-1)/(j-1)$$

2nd Iteration: exploring scoring functions During the second iteration we mainly explored different sentence representations and scoring functions. Using plain text, we measure *ROUGE* scores (Lin, 2004), more precisely R1-F, R2-F, and RL-F. We use vector representations of text based on (i) $tf \cdot idf$ and Latent Semantic Analysis; and (ii) pre-trained French word embeddings from (Fauconnier, 2015), and score sentences based on cosine similarity. Word embeddings are trained with `word2vec` (Mikolov et al., 2013). We experimented with both CBOW and Skip-Gram variants without significant performance differences. Measuring similarities between sliding windows instead of sentences directly was meant to reduce impact of isolated sentences of low similarities. In fact, because our data don't perfectly match, there may be sentences with a very low similarity inside segments that actually discuss the same point. Parameters related to the sliding windows are the window size, and the overlap. We experimented with all combinations of $s \in \{0, 1, 2, 3, 4, 5, 10\}$, $o \in \{1, 2, 3, 5\}$. Related to its scores we consider *aggregation* and *reduction* function as parameters and experiment with $agg \in \{sum, mean, max\}$ and $red \in \{sum; product\}$.

3rd Iteration: fine tuning embedding based models During the alignment phase, we found that the dynamic programming algorithm may keep the same direction for a long time. For example one report sentence may get high similarities with a lot of transcription sentences, resulting in a too monotonical alignment. To limit this behavior, we introduce horizontal and vertical decay factors (respectively hd and vd), typically in $[0; 1]$, that lower scores in the same direction. We then consider a decayed alignment matrix A' such as:

$$A'_{i,j} = A_{i,j} \times D_{i,j}$$
$$D_{i,j} = \begin{cases} D_{i-1,j} \times (1 - hd) \text{ if } A_{i-1,j} > A_{i,j-1} \\ D_{i,j-1} \times (1 - vd) \text{ otherwise} \end{cases} \quad (7)$$

The decay is reset to $D_{i,j} = 1$ at each change of direction.

4th Iteration: `public_meetings` Finally, we select a set of public meetings in order to make it available for reproductions and benchmarks. This smaller corpus is used as a test set: no fine tuning has been done on this data for both the alignment and the summarization tasks.

4.2. Other models

We also consider two linear segmentation baselines, namely TEXTTILING of (Hearst, 1997) and C99(Choi, 2000).

Linear segmentation baselines are penalized in comparison to our methods since they do not use the report document content. In particular, our methods cannot be wrong about the segment number since it is fixed by report side segmentation. Therefore, to make a fairer comparison, we only consider parameters sets that produce the excepted number of segments. Segment number can be explicitly set in C99 whereas we had to grid search TEXTTILING parameters.

GRAPHSEG from (Glavas et al., 2016) has been considered, but producing long enough segments to be comparable with our work requires a low *relatedness threshold*, which exponentially increases the computational cost.

4.3. Summarization

We trained neural summarization models on our data, first using gold set only, then incorporating automati-

cally aligned data. Pre-processing include filtering segments based on their number of words and sentences, i.e. we consider segments if $10 \leq \#words \leq 1000$ and $3 \leq \#sentences \leq 50$.

Using `OpenNMT-py`[2](Klein et al., 2017) we train *Transformer* models (Vaswani et al., 2017) similar to the baseline presented in (Ziegler et al., 2019) with the difference that we do not use any copy-mechanism.

Evaluation is conducted against our `public_meetings` test set and uses the ROUGE-F metric (Lin, 2004).

5. Results

5.1. Automatic Alignment Evaluation

Table 1 compares performances of automatic alignment models.

Diagonal baseline shows interesting performances. In particular, it outperforms by a large margin linear segmentation algorithms and both of our tf-idf and ROUGE based models.

Embeddings based approaches are on a totally different level, with performances twice better than the diagonal baseline, and more than three times better than any other considered algorithm on the validation set.

Introducing decays at the alignment stage is meant to avoid the alignment to be too monotonic. We started experimenting with small decays on both horizontal and vertical axes. Results make it clear that decays are key parameters. In particular, we found *vertical decay* (vd) to have a greater impact, while *horizontal decay* (hd) should be turned-off for maximal performances. Similarly, scaling scores to the power of $p > 1$ during alignment improves every model. In fact, it helps the model to distinguish good scores from average ones.

Sliding windows performs better that sentence representation (i.e. $s = 1, o = 0$) in most case – only tf-idf models reach its top scores without it. However, we observed many different configurations of sizes, overlaps, aggregations and reduction functions reach high scores.

5.2. Human Evaluation

Human annotators align transcription segments with respect to report segments based on a pre-alignment produced by automatic alignment models. As we were fine tuning our models, we provided better pre-alignments, eventually making the annotator's task easier The alignment process for the annotators consists in checking the pre-alignment and correcting mismatches one segment at a time. We report human evaluated segment-level accuracy as the ratio of segments that were not modified by the annotator against the total number of segments.

Figure 3 and table 2 show, for each iteration, the accuracy distribution. We observe that accuracy is consistently increasing over iterations.

[2]`https://github.com/OpenNMT/OpenNMT-py`

	#documents	**Annotator Score ↑** $(mean, median)$
1rst iteration	12	18.63 – 15.73
2rst iteration	88	50.44 – 53.56
3rd iteration	38	57.23 – 55.02
`public_meetings`	22	72.67 – 80.08

Table 2: Human evaluation of automatic alignments

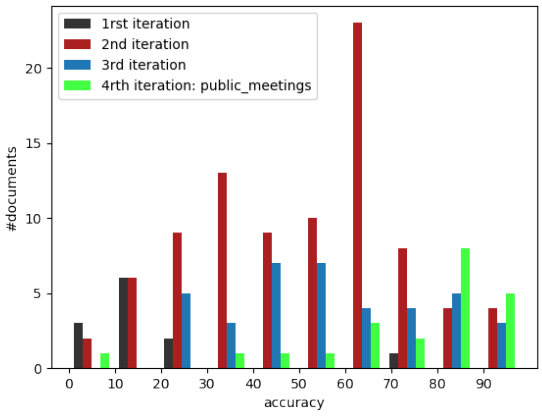

Figure 3: Annotator evaluation with respect to the automatic pre-alignment for each iterations

5.3. Summarization

Summarization models have first been trained on human annotated alignments only, then with a larger dataset that also contains 70,000 more training pairs emanating from automatic alignment. We find that using automatic alignment for data annotation makes a substantial difference in the summarization performance of almost +4 ROUGE points (table 3). This result is encouraging and motivates us to continue automatic annotation.

Dataset	**#Pairs** $(train, test)$	**ROUGE Score (F)** $(R1, R2, RL)$
Gold dataset	21k – 1060	52.80 / 29.59 / 49.49
Gold + Automatic	91k – 1060	56.56 / 35.43 / 53.55

Table 3: Scores on the `public_meetings`test set of automatic summarization models trained on human references only vs. extend the dataset with annotations from automatic alignment

6. Discussion and Future Work

During the alignment process, we make the assumption that each transcription segment must be aligned. However, in practice we asked human annotators to filter out irrelevant segments. Such segments are part of the validation set, but

Model	Window (s, o)	Window Scoring (agg, red)	Alignment (hd, vd, p)	Dev. Acc. (μ-avg) ↑ $(seg.\%, word\%)$	Dev. WD ↓	Test Acc. (μ-avg) ↑ $(seg.\%, word\%)$	Test WD ↓
TextTiling	–	–	–	–	–	09.36 – 06.66	39.61
C99	–	–	–	–	–	05.68 – 05.03	42.49
Diagonal	–	–	–	18.59 – 21.55	17.43	20.75 – 23.28	34.61
tf-idf	2 – 1	sum – sum	0 ; 0 ; 4	5.02 – 5.23	13.80	4.10 – 5.17	23.02
	10 – 3	mean – mul	10^{-4} ; 10^{-4} ; 1	9.69 – 10.90	17.33	9.91 – 10.94	33.00
	1 – 0	max – prod	10^{-4} ; 10^{-4} ; 2	10.93 – 12.33	17.74	10.29 – 10.90	35.05
ROUGE	10 – 2	cat – sum	10^{-4} ; 10^{-4} ; 1	11.43 – 12.70	17.157	9.25 – 9.66	32.99
	2 – 0	cat – sum	10^{-4} ; 10^{-4} ; 4	14.52 – 16.95	17.85	10.63 – 11.76	34.01
Embeddings	2 – 1	sum – prod	0 ; 0 ; 4	45.62 – 54.06	10.81	64.36 –72.95	19.872
	2 – 1	sum – prod	10^{-4} ; 10^{-4} ; 4	60.96 – 70.73	10.94	58.97 – 68.09	23.43
	2 – 1	sum – prod	0 ; 10^{-4} ; 4	**61.00 – 72.50**	**10.38**	**69.36 – 79.06**	**15.09**

Table 1: Automatic alignment models evaluation against the validation set (202 reference meetings) and `public_meetings` test set (22 meetings) on three metrics: segment accuracy, word accuracy, and WindowDiff.

flagged in order that they should not be assigned to any report segments. During evaluation we penalize models for each false alignment assigned to irrelevant segments so that our results are comparable to future models capable of ignoring some transcription segments. To get an idea of how important this phenomenon is, we adapt word accuracy to ignore irrelevant segments and find a 4.7% absolute difference.

$$word\_acc = \frac{\#W_{aligned}}{\#W}$$

$$pos\_word\_acc = \frac{\#W_{aligned}}{\#W - \#W_{irrelevant}}$$

Word embedding vectors used in this work have been trained by (Fauconnier, 2015) who made them publicly available. [3]. While they make our results fully reproducible, training embedding vectors on our data would be an interesting area for future research and could improve the quality of the automatic alignment.

Lastly, we would like to study whether the alignment scores provided by our models could be used to predict the alignment quality. Such predictions could be used to filter automatic annotations and use only the potentially relevant automatically aligned segments.

7. Conclusion

This paper has explored the development of automatic alignment models to map speaker interventions from meeting reports to corresponding sentences of a transcription. Meetings last several hours, making them unsuitable sources for training as-is; therefore, segmentation is a key pre-processing step in neural approaches for automatic summarization. Our models align transcription sentences – as provided by our speech recognition system – with respect to report segments, delimited by tags in the document (either in the header or when explicitly specifying a change of speaker).

We introduce `public_meetings` , a novel meeting summarization corpus against which we evaluate both automatic alignment and summarization.

We have shown that our automatic alignment models allow us to greatly increase our corpus size, leading to better summarization performance on all *ROUGE* metrics (R1, R2, RL).

[3]More information and vectors can be found at `https://fauconnier.github.io/#data`

8. Bibliographical References

Alemi, A. A. and Ginsparg, P. (2015). Text Segmentation based on Semantic Word Embeddings. *CoRR*, abs/1503.0, mar.

Badjatiya, P., Kurisinkel, L. J., Gupta, M., and Varma, V. (2018). Attention-based neural text segmentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10772 LNCS:180–193.

Banerjee, S. and Rudnicky, A. I. (2006). A TextTiling based approach to topic boundary detection in meetings. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, volume 1, pages 57–60.

Barzilay, R. and Elhadad, N. (2003). Sentence alignment for monolingual comparable corpora. In *Proceedings of the 2003 conference on Empirical methods in natural language processing* -, volume 10, pages 25–32, Morristown, NJ, USA. Association for Computational Linguistics.

Beeferman, D., Berger, A., Lafferty, J., Cardie, C., and Mooney, R. (1999). Statistical Models for Text Segmentation. *Machine Learning*, 34(1):177–210.

Braunschweiler, N., Gales, M. J. F., and Buchholz, S. (2010). Lightly supervised recognition for automatic alignment of large coherent speech recordings. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association, INTERSPEECH 2010*, pages 2222–2225.

Choi, F. Y. Y. (2000). Advances in domain independent linear text segmentation. *1st Meeting of the North {A}merican Chapter of the Association for Computational Linguistics*.

Fauconnier, J.-P. (2015). French word embeddings. `http://fauconnier.github.io`.

Gehrmann, S., Deng, Y., and Rush, A. M. (2018). Bottom-Up Abstractive Summarization. *Proceedings of EMNLP*.

Glavas, G., Nanni, F., and Ponzetto, S. P. (2016). Unsupervised text segmentation using semantic relatedness graphs. In **SEM 2016 - 5th Joint Conference on Lexical and Computational Semantics, Proceedings*, pages 125–130, Berlin, Germany. Association for Computational Linguistics.

Hearst, M. A. (1997). TextTiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23(1):33–64.

Hermann, K. M., Kočiský, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. (2015). Teaching Machines to Read and Comprehend. In *Advances in Neural Information Processing Systems*, volume 2015-Janua, pages 1693—-1701, jun.

Hernandez, F., Nguyen, V., Ghannay, S., Tomashenko, N., and Estève, Y. (2018). TED-LIUM 3: Twice as Much Data and Corpus Repartition for Experiments on Speaker Adaptation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11096 LNAI, pages 198–208, may.

Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. M. (2017). OpenNMT: Open-source toolkit for neural machine translation. *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of System Demonstrations*, pages 67–72.

Lecouteux, B., Linarés, G., and Oger, S. (2012). Integrating imperfect transcripts into speech recognition systems for building high-quality corpora. *Computer Speech and Language*, 26(2):67–89.

Lin, C. Y. (2004). Rouge: A package for automatic evaluation of summaries. *Proceedings of the workshop on text summarization branches out (WAS 2004)*, pages 25–26.

Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, Ł., Shazeer, N., Kaiser, L., Shazeer, N., J. Liu, P., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., and Shazeer, N. (2018). Generating Wikipedia by Summarizing Long Sequences. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, jan.

Meignier, S. and Merlin, T. (2010). LIUM SPKDIARIZATION: AN OPEN SOURCE TOOLKIT FOR DIARIZATION. *CMU SPUD Workshop*, page 1433518.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations ofwords and phrases and their compositionality. In *Advances in Neural Information Processing Systems*.

Nallapati, R., Zhou, B., dos Santos, C. N., Gulcehre, C., Xiang, B., dos Santos, C., and Xiang, B. (2016). Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond. *Proceedings of CoNLL*.

Nelken, R. and Shieber, S. M. (2006). Towards robust context-sensitive sentence alignment for monolingual corpora. In *EACL 2006 - 11th Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference*, pages 161–168.

Paulus, R., Xiong, C., and Socher, R. (2017). A Deep Reinforced Model for Abstractive Summarization. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, may.

Pevzner, L. and Hearst, M. A. (2002). A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36.

See, A., Liu, P. J., and Manning, C. D. (2017). Get To The Point: Summarization with Pointer-Generator Networks. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, apr.

Song, Y., Mou, L., Yan, R., Yi, L., Zhu, Z., Hu, X., and Zhang, M. (2016). Dialogue Session Segmentation by Embedding-Enhanced TextTiling. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 08-12-Sept:2706–2710, oct.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., Polosukhin, I., Kaiser, L., and Polosukhin, I. (2017). Attention is All you Need. In I Guyon, et al., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., jun.

Ziegler, Z. M., Melas-Kyriazi, L., Gehrmann, S., and Rush, A. M. (2019). Encoder-Agnostic Adaptation for Conditional Language Generation. *CoRR*, abs/1908.0.