SmartCaptions: A fast, light-weight system for subtitle segmentation

Anonymous

Abstract

Subtitles make videos more accessible to a wider audience, including foreign-language speakers, hard-of-hearing individuals, and anyone who can not watch a video with sound. The recent emerging of online video sharing and work-from-home, study-from-home demands automatic solutions for video transcribing as well as subtitle segmentation. In this paper, we describe a subtitle segmentation system based on the rule-based approach which uses very few resources: a constituent parser. The system uses no training data for this task. The system is simple and was developed in a short period of time. Since it does not employ any additional resources or any sophisticated machine learning methods, it still delivers some errors. However, it could be considered as a baseline system for the task. On the other hand, it shows what can be obtained using a simple rule-based approach and presents a few situations where the rule-based approach can perform well without a machine learning model.

Introduction

Automatic subtitling has been growing rapidly due to the large amount of content produced by the TV channels and the entertainment industry. The recent exploding demand for video usages such as online video sharing, online learning, and the work-from-home movement has created an enormous interest in automatic subtitling. The effort has mainly focused on speech recognition in the first step due to the nature of the task. However, there is an increasing demand for improvement from the viewer in the quality of the displayed subtitles.

The quality of the displayed subtitles relates to three important cognition processes including vision cognition, acoustic cognition, and semantic cognition. For the visual perspective, subtitle position, subtitle and its background colors, transition effects. For acoustic cognition, displaying duration and alignment are extremely important because misalignment between subtitle and audio creates an adverse effect on the focus of the audience to the content. Finally, semantic cognition relates to some factors such as capitalization, segmentation, and the use of punctuation, as well as the use of acronyms and numbers. Among these factors, psycholinguistics literature has addressed proper segmentation as the key factor in readability []. The overall requirement is that the subtitle should be segmented in favor of readability. It should minimize the cognitive effort to correct the poorly segmented text. Figure 1 shows a bad subtitle segmentation on a live stream video on Behance.net website.

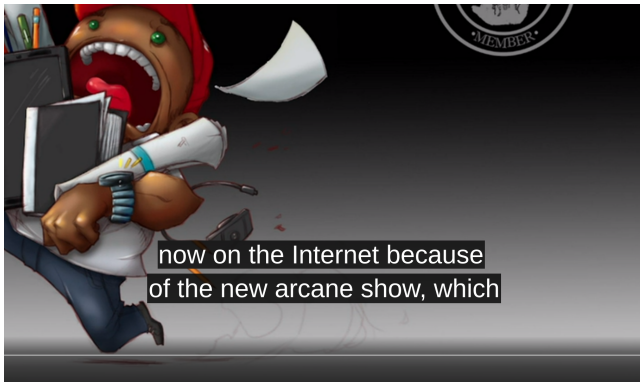

Figure 1: Screenshot of a bad subtitle display on Behance.net. The sentence is "*Like being blown up right now on the Internet because of the new arcane show, which you guys haven't watched.*". It should be displays as following segments: "*Like being blown up*","*right now on the Internet*","*because of the new arcane show,*","*which you guys haven't watched.*",

The subtitle segmentation relates closely to the cognitive process of which people read text. According to psycholinguistic literature, readers favor syntactic information: grouping words into syntactic phrases and clauses. Hence, in order to facilitate readability, the text should be segmented according to coherent linguistic breaks. Reading subtitles is similar to reading any other text materials for which segmentation that is not based on syntactic coherence can reduce readability. From a linguistic perspective, coherent line breaks can be created using syntactic information. As such, subtitle segmentation can be done by analyzing syntactic information of the text.

In this paper, we argue that subtitle segmentation can be done using a parse tree because a parse tree provides syntactic information. A parse tree also presents its information in a hierarchical order of the elements in the sentence that

perfectly suits the subtitle segmentation task. Fortunately, thanks to the recent advance in natural language processing, syntactic parse tree has been produced from raw text at a very high accuracy [5], which we believe it is good enough to serve our purpose in this task. In particular, we present an open-source, fast, light-weight solution for subtitle segmentation. First, our rule-based system depends on a single high-accuracy open-source syntax tree parser. As a rule-based system, it requires no training data, which is an extremely advantage against other machine learning-based system because it can be highly adapted to other languages without any data collections. Secondly, our system can be easily modified to adapt to the specific requirements of length limit and writing style of the industry. This means the subtitle can be displayed at its best that fits the displaying devices such as computer monitor, TV, and mobile phone.

Related Work

Prior methods for subtitle segmentation make use of a classifier to predict break point. The early approach used Support Vector Machine and Logistic Regression classifiers to predict correct/incorrect segmented subtitles [1]. Conditional Random Field classifier has been used for the same task with distinction between line breaks and subtitle break Alvarez et al. [2]. Recently, Long-Short Term Memory Network (LSTM) is use to to insert periods to improve the readability of automatically generated Youtube captions. However, these methods did not focus on length constraint required by the industry standard. Liu, Niehues, and Spanakis, on the other hand, proposed to enhance ASR output by incorporating transcription and text compression to generating more readable subtitles. A transformer-based model to improve the readability of subtitle on movies [4]. However, these methods requires training data and training a learning model to perform the task, which is very costly and improper for industry.

Method

Input: Our system accepts transcribed text (i.e. ASR text) as the input. The text has to be split into sentences (e.g. by period, question, exclamation markers). The system works much better if the sentences have commas to separate syntactic clauses and noun phrases.

Tree parser: We use only a single linguistic resource in this system: the constituent tree parser. There have been many publicly accessible constituent tree parsers such as Stanford Parser and SpaCy. However, we choose SpaCy because it can be used in commercial products. We use SpaCy with the Berkeley Neural Parser [5]. Specifically, we use Spacy version 3 and the "en_core_web_trf" package for the highest accuracy. The ASR text is passed through the SpaCy to generate its syntax tree without any further preprocessing. We obtained the SpaCy output in Treebank's format. Finally, we parse this into a JSON object which will be used in the system.

Since the constituent tree is presented in a hierarchical form of S + VP in which most of the cases, the VP can be expanded into a large subtree. If we directly separate S and VP, it will create many incorrect breaks such as break right after the pronoun. As such, we have to modify the tree structure so that it can break the VP into smaller chunks whose first chunk can go together with the S.

Move punctuation: In a sentence with punctuation, the punctuation should always stick to the previous token (denoted as P). However, in the parse tree, they are usually represented by a child node of the same levels as the phrases/clauses on the left and the right (in case the marker is a comma). Hence, the first rule is to merge the P node and the punctuation node.

Move contracted forms: In English, contracted forms are commonly used, however, once being tokenized, the contracted forms usually being split into multiple tokens, for example "*I*" and "*'m*". Similar to the punctuation, the second token, which goes with the apostrophe, has to stick to the first part. We merge the second token, into the node of the first token.

Collapse atomic clauses: A clause is usually represented by a big subtree. We define an atomic clause as a clause that can fit into the length limit of the subtitle. Hence, we collapse the subtree that represents an atomic clause and then we replace it with a single node with the text of the clause. We collapse all clausal subtrees whose roots are labeled as *S, SINV, SBAR, SBARQ, SQ* [3].

Collapse atomic phrases: Similar to the atomic clause, we define an atomic phrase as a phrase that can fit into the length limit of the subtitle. Among 21 types of phrases defined by the Treebank [3], we collapse almost all of the types except verb phrase (VP) because in many cases, if we collapse VP, the pronoun would be separated from the VP creating a type 8 error (presented in the error analysis). The complete list of phrases, which are collapsed, consists of *ADJP, ADVP, CONJP, LST, NAC, NX, PP, PRN, PRT, QP, RRC, UCP, NP, WHADJP, WHAVP, WHNP, WHPP, X, FRAG, INTJ*[3].

Collapse by comma: Once all atomic phrases and clauses are collapsed, the sentence is split into chunks such that no chunk exceeds the length limits. A chunk terminates if it has a token ending with a comma.

Collapse short chunks: The previous step leaves many short chunks, for example, a sentence "*Yeah, Yeah, I see your point*" will be split into three chunks by commas: "*Yeah*", "*Yeah,*", and "*I see your point*". The last step is to merge consecutive short chunks into longer chunks if it is possible. As such the first two chunks of the mentioned sentence will be merged into one, leaving a subtitle segmentation as follow:

Yeah, Yeah,
I see your point

Error analysis

To our best knowledge, there is no subtitle segmentation dataset for evaluating models in this task. We instead run a human error analysis. We follow the list of errors published by Ted Talk[1]. It contains 10 types of segmentation

[1]https://translations.ted.com/English_Style_Guide

#	Error type	#errors
1	To be	1
2	Complex form	10
3	Contracted form	1
4	Infinitive verb	0
5	Article-noun	0
6	There + be	0
7	Relative pronoun + clause	8
8	Pronouns as a subject	5
9	Determiners, adjectives, etc	0
10	Prepositions	3
11	Missing breaks	10
	Total	38 (5.34%)

Table 1: Error analysis of 711 subtitle lines generated by our systems.

errors. We add the eleventh error type that the sentence is not punctuated at the appropriate position, which leads to a bad following break. The details of the error types are presented as follows. The strikethrough indicates the incorrect segmentation.

1. Break the verb "to be" and the predicate ("~~*Jack is/a girl*~~")
2. Break complex grammatical forms ("~~*going/to*~~")
3. Break lines after contracted forms of verbs ("~~*Remember that book? It's/here*~~")
4. Break the "to" infinitive verb ("~~*It's not difficult to/eat slowly*~~")
5. Keep articles and nouns together ("~~*Paris is a/city in France*~~")
6. Break "there" + "be" ("~~*I heard there/is*~~ ")
7. Break relative pronouns and the clause they introduce
8. Separate a pronoun used as the subject of a clause from the verb/component
9. Break the line or subtitle after determiners: adjectives, numerals, demonstratives (like this or those), possessives (like his or the dog's) or quantifiers (like some, any, every, a lot of, etc.)
10. Break after a preposition (in, on, under, etc.) followed by the noun they refer to.
11. Lack of break line at the appropriate position.

We analyze a subtitle of a live streaming video on Behance.net. After segmentation, the subtitle results in 711 lines. Table 1 reports the error break-down. Out of 711 segments, 38 segments are wrongly terminated, accounting for 5.34% percent. Among the error types, breaking complex grammatical forms and lacking appropriate breaking are the most common errors with 10 errors for each type. Breaking relative pronoun and its clause (8 errors) are the third highest error type. The system makes no errors in four categories including breaking after contracted forms, breaking a "to" infinitive verb, breaking article and noun, and breaking after adjectives, numeral, and so on. Figure 2 presents a few outputs of our system on long sentences.

Discussion

Error sources: The error analysis gives us more insight into which errors appear the most, it also gives us more insight into why the errors appear. As our observation, there are several reasons that created those errors:

- Spontaneous nature of live streaming conversation: Live streaming conversation is not well prepared as a formal speech or written text. As such, it contains a lot of noise such as verbal pause, duplicate words, and grammatically incorrect sentences. As such, it might severely affect the performance of the tree parser.
- Word errors and punctuation errors in ASR text: As the texts are automatically generated using speech recognition technology, they contains a certain amount of word errors and low-quality punctuation restoration. Even a single incorrect commas might ruins the parse tree because the existing tree parsers are incapable of dealing with noise this kind of noise in text.
- Incorrect parse tree: Even though tree parser has get very high accuracy in syntax parsing, we can still observe errors in low-noise sentences due to the imperfect of the parser. The error will propagate to the performance of the segmentation model.

Applications: Since there is no existing dataset for subtitle segmentation especially for ASR text on live streaming platform. We believe that our system is a good baseline for later studies. One can also use this system to generate a large-scale subtitle segmentation dataset to facilitate the study in this topic.

Conclusion

In this paper, we present a fast, light-weight, rule-based system for subtitle segmentation. We show that syntax tree can be effectively used in a rule-based system to segment subtitle at any length limit with a low error rate. We discuss the sources of errors in our system that may suggest future research to improve our system. We believe that this is a good baseline for future studies for subtitle segmentation. We suggest this can be use for data generation in machine-learning-based approach for subtitle segmentation.

References

[1] Álvarez, A.; Arzelus, H.; and Etchegoyhen, T. 2014. Towards customized automatic segmentation of subtitles. In *Advances in Speech and Language Technologies for Iberian Languages*, 229–238. Springer.

[2] Alvarez, A.; Martínez-Hinarejos, C.-D.; Arzelus, H.; Balenciaga, M.; and del Pozo, A. 2017. Improving the automatic segmentation of subtitles through conditional random field. *Speech Communication*, 88: 83–95.

[3] Bies, A.; Ferguson, M.; Katz, K.; MacIntyre, R.; Tredinnick, V.; Kim, G.; Marcinkiewicz, M. A.; and Schasberger, B. 1995. Bracketing guidelines for Treebank II style Penn Treebank project. *University of Pennsylvania*, 97: 100.

```
And I'll be starting
about in one minute or two,
and we'll go from there, right?
--------------------------------
So, um, with this,
I'm going to be starting
on a brand new image.
--------------------------------
I i'm going to,
this is going to be
more like a poster,
almost like an advertisement,
advertisement, say that.
--------------------------------
So I've been a bit of a loss
for words that past couple days
with everything going on.
--------------------------------
But I, you know,
I wanted to create something
that meant a lot to me,
but also,
at the same time is appropriate
with everything that's going on,
especially something
that is a very sensitive topic.
```

```
But I wanted to create
something that is relevant,
especially with everything
that's going on with            //Error
black lives matter and          //Error
racial justice
in the United States, as well as
along with police brutality.
--------------------------------
So I'm going to place
a couple images on here,
and then we're going to kind of //Error
mix them up in a kind of make   //Error
something new out of this.
--------------------------------
I'm liking that in all this I'm //Error
actually going to be using
just a mouse, 'cause I want,
I don't think necessary you     //Error
always have to use
a Wakeham tablet to create
something unique.
```

Figure 2: Examples of the output of our system. The 32-character dash line separates sentences and indicates 32-character limit. The left column presented correct segmentation. The right column shows incorrect segmenation examples with comments indicating the incorrect lines.

[4] Karakanta, A.; Negri, M.; Turchi, M.; Kessler, F. B.; and Povo, T.-I. 2020. Point Break: Surfing Heterogeneous Data for Subtitle Segmentation. In *CLiC-it*.

[5] Kitaev, N.; and Klein, D. 2018. Constituency Parsing with a Self-Attentive Encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2676–2686. Melbourne, Australia: Association for Computational Linguistics.

[6] Liu, D.; Niehues, J.; and Spanakis, G. 2020. Adapting end-to-end speech recognition for readable subtitles. *arXiv preprint arXiv:2005.12143*.