Towards Customized Automatic Segmentation of Subtitles

Aitor Álvarez, Haritz Arzelus, Thierry Etchegoyhen

Human Speech and Language Technologies, Vicomtech-IK4, San Sebastián, Spain
{aalvarez, harzelus, tetchegoyhen}@vicomtech.org

Abstract. Automatic subtitling through speech recognition technology has become an important topic in recent years, as increasingly larger volumes of audiovisual content need to be subtitled and speech recognition can provide the backbone of a solution to tackle this need. Most of the work in this area has focused on improving core speech technology and the related front- and back-end modules to obtain better recognition results. However, subtitling quality also depends on other parameters aimed at favoring the readability and quick understanding of subtitles, like correct subtitle line segmentation. In this work, we present an approach to automate the segmentation of subtitles through machine learning techniques, allowing the creation of customized models adapted to each subtitling company's segmentation rules. Support Vector Machines and Logistic Regression classifiers were trained over a reference corpus of subtitles manually created by professionals and used to segment the output of speech recognition engines. We describe the performance of both classifiers and discuss the merits of this approach for the automatic segmentation of subtitles.

Keywords: automatic subtitling, subtitle segmentation, machine learning

1 Introduction

Automatic subtitling has recently attracted the interest of the speech and natural language processing research communities, notably after the adoption of new audiovisual legislation by the European Parliament in 2007. This legislation regulates the rights of people with disabilities to be integrated in the social and cultural life of the Community, through accessible audiovisual contents by means of sign-language, audio-description and subtitling. As a result, the demand for automatic subtitling has grown rapidly, with public and private TV channels moving to produce subtitles for larger volumes of their content. The effort has focused on quantity in a first step, in order to match legislative requirements, but there is an increasing demand for an improvement in the quality of automatically generated subtitles as well.

The quality of subtitles involves several parameters linked to subtitle layout, duration and text editing. Layout parameters include: the position of subtitles on screen; the number of lines and amount of characters contained in each line; typeface, distribution and alignment of the text; colors for front and background; different colors per speaker; and transmission modes, i.e. blocks or scrolling/word-by-word. Duration parameters involve delay in live subtitling and the persistence of subtitles on screen.

Finally, text editing parameters are related to capitalization and punctuation issues, segmentation and the use of acronyms, apostrophes and numerals.

Among these quality features, the strong need for proper segmentation is supported by the psycholinguistic literature on reading [1], where the consensual view is that subtitle lines should end at natural linguistic breaks in order to favor readability and minimize the cognitive effort produced by poorly segmented text lines [2].

In order to address the need to tackle subtitle quality aspects beyond bare speech recognition and to provide solutions adaptable to the standard guidelines and specific rules of companies, we explored a flexible approach based on machine learning techniques and tested it on the automated segmentation task. Specifically, we trained Support Vector Machines and Logistic Regression classifiers on subtitle corpora created by professional subtitlers and used the resulting models to filter and select optimal segmentation candidates. The results we present involve the use of these two classifiers for the automatic segmentation of subtitles in Basque, although the approach is not language-specific as it only requires properly segmented training material of the type created by subtitling companies under their own guidelines.

This processing pipeline for segmentation has been integrated into the automatic subtitling system described in [3], taking the output of speech processing engines to provide customized segmented subtitles.

The paper is structured as follows. Section 2 describes existing solutions and studies regarding automatic subtitling and segmentation. Section 3 looks at standard issues and considerations for the segmentation of subtitles. Section 4 describes the machine learning approach we implemented. Section 5 presents the experiments and evaluation results. Finally, Section 6 draws conclusions and describes future work.

2 Related Work in Automatic Subtitling

There is extensive research focused on automatic subtitling, mainly through the using of Automatic Speech Recognition technology for the recognition and alignment tasks [4][5][6][7]. Most of the work in the area has centered on improving recognition accuracy and producing well-synchronized subtitles. Audimus [8] is a reference system in the field, as it provides a complete framework for automatic subtitling of broadcast news contents with low error rates in both batch and live modes. It includes an automatic module for subtitle generation and normalization aimed at improving readability. This module includes conversion processes (e.g., numbers to their digit representation), capitalization, punctuation, and color switching based on speaker gender detection. However, segmentation was performed minimally, using only information about the maximum amount of characters permitted per line. The Audimus system was improved and extended to several languages within the European project SAVAS[1]. Many quality features were considered in the development of the new systems for automatic subtitling, but technology for automatic segmentation has not been developed so far.

[1] http://www.fp7-savas.eu/

Although considerable importance is commonly placed on most of the quality parameters described above, proper line-breaking has generally been disregarded [9]. A survey of the literature in the field actually provides no references on the topic of automatically segmenting subtitles. A few studies were carried out on related topics, as can be found for instance in [9], which explores the way line-breaking is commonly performed, and in [2] which studies the impact of arbitrary segmented subtitles on readers. The importance of segmentation has been noted by [10], a study whose aim was to verify whether text chunking over live re-spoken subtitles had an impact on both comprehension and read speed. They concluded that even though significant differences were not found in terms of comprehension, a correct segmentation by phrase or by sentence significantly reduced the time spent reading subtitles.

3 Subtitle Segmentation

3.1 Standard Guidelines

A number of guidelines for subtitling have been published over the years. Among well-known ones are: Ofcom's Guidance on Standards for Subtitling[2]; BBC's Online Subtitling Editorial Guidelines[3]; ESIST's Guidelines for Production and Layout of TV Subtitles[4], the Spanish UNE 153010 norm [11] on subtitling for the deaf and hard of hearing and a reference textbook on generally accepted subtitling practice published in 2007 by Jorge Diaz-Cintas and Aline Remael [12]. Standard guidelines cover the various aspects of subtitle quality, such as subtitle segmentation, and standard practices along these recommendations are shared among subtitling companies and broadcasters.

In terms of segmentation, all standard recommendations conclude that it must benefit and improve readability. For this purpose, considering syntactic information to create linguistically coherent line-breaks is the preferred and most adopted solution in the community. This follows from results in psycholinguistic research, which show that readers analyze texts in terms of syntactic information [13], grouping words corresponding to syntactic phrases and clauses [14]. Reading subtitles is a similar task and subtitles for which segmentation is not based on coherent syntactic groups can thus be assumed to trigger sub-optimal reading [15]. In order to facilitate readability, subtitle lines should thus be split according to coherent linguistic breaks, and the generally accepted solution is to operate the splits at the highest possible syntactic node. This ensures that fragments split along these lines encompass the largest possible amount of related semantic information.

[2]http://www.ofcom.org.uk/static/archive/itc/itc_publications/codes_guidance/standards_for_subtitling/subtitling_1.asp.html

[3]http://www.bbc.co.uk/guidelines/futuremedia/accessibility/subtitling_guides/online_sub_editorial_guidelines_vs1_1.pdf

[4]http://www.translationjournal.net/journal/04stndrd.htm

3.2 Issues in Automatic Segmentation

Although the strong need for proper segmentation and the general constraints that apply to it are clear, there are issues regarding the implementation of automated segmentation.

First, as the previously described guidelines are fairly general in terms of what constitutes a proper subtitle split, there is actual variation among professional subtitlers when it comes to executing the actual segmentation. These variants are usually reflected as distinct sets of company-specific rules, which makes a generic automated solution all the more difficult to achieve as such a solution would have to either disregard company-specific rules or require resource-consuming adaptation of the syntactic rule sets on a case by case basis.

Secondly, the automatic detection of the highest syntactic node requires language processing tools for sentence analysis. For major languages like Spanish or English, several such tools are available, e.g. Freeling [16], OpenNLP [17] or the parsers developed by the Berkeley [18] and Stanford [19] groups. For other languages, particularly under-resourced ones, there can be a lack of robust natural language analyzers, which would limit the possibilities of using a syntax-based approach for segmentation.

Finally, a correct syntactic analysis and detection of the highest nodes in subtitles does not guarantee proper segmentation. Several other features have to be considered simultaneously, such as the amount of characters, timing issues and, as previously mentioned, the specific splitting rules used by each subtitling company. All these features have a clear impact on proper subtitle segmentation and need to be taken into account for each specific subtitle.

An ideal solution for the automatic segmentation of subtitles would thus have to (1) correspond to the specific rules used by each subtitling company, and (2) simultaneously consider all relevant information like character sequence length and timing.

In the remainder of the paper, we present a possible solution that involves the use of machine learning classifiers to create segmentation models adapted to each company's needs, thus providing a highly customizable and language-independent solution. This approach has the additional advantage of allowing the simultaneous integration of different features to reach optimal segmentation.

4 Machine Learning for Automatic Segmentation

This section describes the core components of the machine learning approach we followed. We define the automatic segmentation problem as a binary classification task, where subtitles with correct or incorrect segmentation are split into two classes.
Positive ("correct") feature vectors were extracted from professionally-created subtitle data and contain the segmentation marks found in the corpus; negative ("incorrect") vectors were generated by automatically inserting improper segmentation marks. Classifiers were then trained on balanced sets formed with these two types of vectors and used for the segmentation task.

4.1 Corpus Characteristics

The corpus used to train and test the classifiers was composed of subtitles that were manually created by professional subtitlers for TV cartoon programs in Basque, for a total amount of 158011 subtitles. The files were provided in SRT format, indicating start and end times codes for each subtitle and presented in blocks of a maximum of two lines. The corpus was split between training and test sets containing 80% and 20% of the data, respectively.

The subtitles in the corpus were manually generated considering subtitle layout, duration and text editing features. In particular, the segmentation rules followed by the subtitlers focused on maintaining linguistic coherence, splitting subtitles according to the highest possible syntactic node.

4.2 Corpus Processing

In order to train classifiers, both positive and negative examples are necessary, from which to extract feature vectors suitable for the task. We thus prepared a balanced set of positive and negative sets by transforming the original subtitles into a task-specific format. Positive examples were generated by merging consecutive lines in reference subtitles into a single sentence containing the original segmentation mark. Each such transformed sentence was then used as a basis to generate a set of negative examples, by moving the original correct segmentation symbol to other positions in the sentence. All possible negative training examples were generated in a first step, each with a different segmentation point, and a randomly selected subset of these possible incorrect examples was used to balance the amount of positive and negative training elements.

Table 1 provides examples of transformed subtitles, including positive and negative candidates.

Reference subtitles	*Transformed data (examples)*	*Label*
1	Lapitza eta erregela. #S# Ez dira berdinak	Correct
00:00:47,430 → 00:00:49,448	Ez dira berdinak, #S# ezta pentsatu ere.	Correct
<S1>Lapitza eta erregela.	Lapitza #S# eta erregela. Ez dira berdinak	Incorrect
2	Lapitza eta #S# erregela. Ez dira berdinak	Incorrect
00:00:51,283 → 00:00:54,660	Lapitza eta erregela. Ez #S# dira berdinak	Incorrect
<S2>Ez dira berdinak,	…	
ezta pentsatu ere.		

Table 1. Training data. The #S# mark denotes a segmentation symbol. The <S1> and <S2> marks correspond to speaker information marks.

Training sentences were thus composed of two parts corresponding to each line in a subtitle and divided by the #S# symbol. Features were computed on each of the parts and on the entire sentence as well. Each feature vector was then categorized with the corresponding label.

4.3 Feature Vectors

The features extracted from the transformed data can be divided into four types of characteristics related to (1) timing, (2) number of characters, (3) speaker change and (4) perplexity as given by a language model built over the training data. A feature vector was calculated for each of the sentences in the transformed data and used to train machine learning classifiers.

The timing feature involved the time difference between the first and second parts of each sentence in the transformed data. It was calculated from the start time of the first word of the second part and the end time of the last word of the first part. Since the reference subtitles provided just the time codes of the first and last words at the subtitle level, a forced alignment algorithm was applied to obtain the start and end times for all the words. For example, no time code was originally available for the first word of the second sentence in the second subtitle ("ezta") shown in Table 1, a necessary piece of information for our approach. To extract the missing time-codes, the forced alignment system for Basque presented in [3] was used.

To characterize aspects related to the number of characters, three features were calculated from the transformed data. The first two contained the amount of characters of the first part and second part of each sentence, respectively, and the third feature indicated the total number of characters in the entire (bi-)sentence.

Speaker change information was available in the reference subtitles and converted into a Boolean value: speaker changes were defined to have value 1 if true and 0 otherwise.

The last feature indicated the perplexity value given by a language model built on the correct sentences in the transformed data. Given that Basque is a morphologically rich language and considering the scarcity of the training data, the language model was built using Parts-Of-Speech (POS) information. For this end, the Eustagger [20] toolkit was used, which includes a morphological analyser and a POS tagger for Basque developed by the IXA group of the University of the Basque Country. An unpruned 9-gram language model was estimated using the KenLM toolkit [21], with modified Kneser-Ney smoothing [22]. The average perplexity value was 24.25 on the test set.

4.4 Segmentation Algorithm

As mentioned above, the automatic segmentation module was integrated into our automatic subtitling system for Basque. This system produces alignments between audio signal and transcripts, thus producing time-codes for each word and providing a basis for the complete generation of subtitles. The segmentation module benefits from the automatically aligned and time-coded words to create candidates for segmentation. These candidates are then measured against the machine learned models and optimal candidates selected according to the score obtained by their feature vectors. The algorithm for candidate generation and selection is given in pseudo-code and described below.

```
function add_segmentation()
max_length = maxline_characters * 2; //max length of 2 subtitle lines
imin = 0; imax = 1; //initialize indexes
words = get_words(); //string of words
CONTINUE:
  while (length (candidates) <= max_length)
  {
    (candidates,...) = generate_candidates(words,imin,imax,...);
    imax++;
  }
  if (exist_valid_candidates())
  {
      (best_i_cut,best_i_max,...) = get_best_candidate(...);
      insert_cut (best_i_cut);
      imin = best_i_cut + 1; imax = best_i_max; //refresh indexes
      right_block = words(imin..imax); //second segment of the cut
  }
  else //no valid candidates
  {
    if length(words(right_block)) > 1 //second part of previous cut
      insert_cut (best_i_max);
      imin = best_i_max + 1; imax = imin + 1;  //refresh indexes
    else
      imin++; imax = imin + 1;
  }
  goto CONTINUE
end // function
```

The function `add_segmentation()` is the entry point for the generation and selection of segmentation candidates. Processing of the text to be segmented is iterative, with validated insertion points taken as new starting points for further processing of the remainder of the text. In other words, we compute segmentation points through short windows of text and repeat the process on the yet unprocessed text after an optimal segmentation has been found for the current window.

Potential points of segmentation are inserted between sequences of consecutive words, where the sole constraint is a maximum allowed sequence length before and after segmentation points. That is, neither sequence on either side of a potential segmentation point can have more characters than this fixed value, which is computed at the beginning of the process and comes from the maximum length in characters for a subtitle line, as observed in the training data.

The candidates are created through the sub-routine `generate_candidates()`. The initial candidates correspond to all combinations of the current sequence of words and the segmentation points. Each candidate thus includes only one segmentation point and the set of all candidates covers the space of potential segmentation points for the current window of words. Feature vectors are then extracted for each candidate and classified according to the previously trained models. In order to reduce the list of current candidates into a more manageable set, we only retain candidates with a model-predicted probability above a fixed threshold. Empirical determination for the task at hand yielded a fixed value of 0.7 for this threshold.

The sub-function `exist_valid_candidates()` checks for the existence of any valid candidate in the filtered set. If the test is positive, the best candidate is selected through the sub-routine `get_best_candidate()`, which returns the candidate with the highest probability according to the model. In case of a tie, the longest candidate is selected. The sequence of words to the right of the last segmentation point is then stored for the next iteration. As this sequence has already been determined to be a an autonomous sequence at this point, it is taken as an indivisible block in the next iteration, i.e. no new segmentation points can be set between the words that compose it.

If `exist_valid_candidates()` indicates no candidates at all, the last stored sequence is considered. If this sequence consists of more than one word, a segmentation point is inserted by default; if it contains just one word, the case is taken to be identical to processing the first word of the text: new sequences and candidates are generated from the sequences that include this word and the next ones.

5 Experiments and Evaluation

Several experiments were carried out using Support Vector Machines (SVM) and Logistic Regression (LR) classifiers using the LibSVM [23] and Scikit-learn [24] toolkits respectively. For the SVM classifier, after testing and comparing different combinations of Kernel functions and methods to perform multi-class classification, the Radial Basis Function (RBF) kernel with nu-support vector classification (nu-SVC) algorithm was selected, as this setup gave the best results. The LR classifier was trained through Stochastic Gradient Descent (SGD).

Table 1 presents results in terms of F-1 measure for each of the test files for both classifiers.

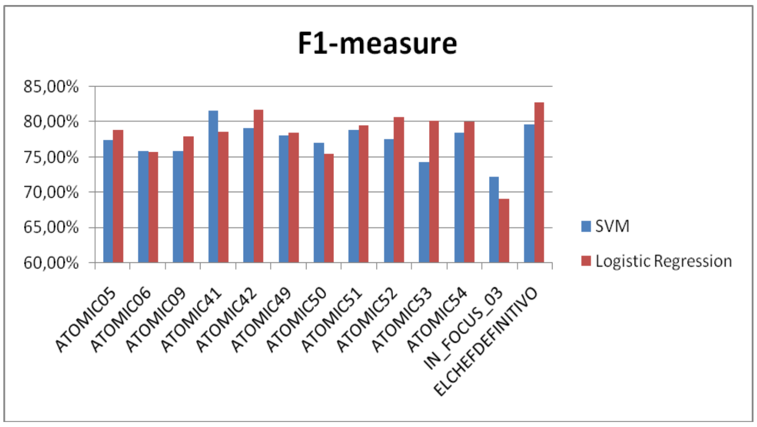

Fig. 1. Segmentation accuracy using SVM and LR classifiers

The results demonstrated similar performance for the two classifiers, with an average score of 74.71% and 76.12% for the SVM and LR classifiers, respectively. In terms of Precision and Recall, the SVM classifier obtained scores of 82% and 69%. In contrast, the LR classifier achieved a Precision of 85% and a Recall of 69%. Interestingly, both classifiers reached identical Recall, showing that on average almost seven out of ten segmentation points are correctly identified in this approach. Combining this result with precision scores above 80%, the general approach can be seen as promising.

6 Conclusions and Future Work

We presented a novel approach to automatic subtitle segmentation which generates and selects optimal segmentation points according to the predictions made by machine learning classifiers. This method provides a customized solution to company-specific segmentation guidelines and rules, as the models are strictly induced from existing segmented corpora and generate similar segmentation on new input.
Additionally, the approach fills a void as far as generating quality subtitles is concerned, given that automatic subtitle segmentation, which is a crucial quality feature, has been somewhat neglected in the research community.
Finally, the method offers a versatile solution as it permits the addition of new features to further tune and improve classification models and subsequent segmentation accuracy.
The preliminary results we have presented are quite satisfactory, with an average recall of nearly 70% and precision above 80% on the test set of a professionally-created corpus of TV cartoon programs in Basque.
In future work, we will pursue experiments on additional corpora, to further evaluate the approach with different domains. More languages will also be tested, as different linguistic characteristics can have an impact on segmentation results, notably in terms of language-dependent infelicitous line endings. We will also explore the impact of including additional features to train classifiers, and evaluate the performance of different feature sets. For instance, incorporating perplexity scores from additional language models trained on surface forms and morphemes might prove beneficiary, as the models would thus include a measure of superficial linguistic knowledge which can be assumed to further improve the proper segmentation of subtitles.

7 References

1. d'Ydewalle, G., Van Rensbergen, J.: Developmental studies of text-picture interactions in the perception of animated cartoons with text. In: Mandl, H., Levin, J. R. (eds.), Knowledge acquisition from text and pictures, pp. 233–248. Elsevier Science (1989).
2. Perego, E., Del Missier, F., Porta, M., Mosconi, M.: The cognitive effectiveness of subtitle processing. Media Psychology, 13, 243–272 (2010).
3. Álvarez, A., Arzelus, H., Ruiz, P.: Improving a long audio aligner through phone-relatedness matrices for English, Spanish and Basque. In: 17th International Conference on Text, Speech and Dialogue (2014).
4. Álvarez, A., Del Pozo, A., Arruti, A.: APyCA: Towards the Automatic Subtitling of Television Content in Spanish. In: Proceedings of the International Multiconference on Computer Science and Information Technology, pp. 567-574 (2010).
5. Bordel, G., Nieto, S., Peñagarikano, M., Rodríguez-Fuentes, L. J., Varona, A.: A simple and efficient method to align very long speech signals to acoustically imperfect transcrip-

tions. In: 13th Annual Conference of the International Speech Communication Association, INTERSPEECH, Portland, Oregon (2012).

6. Driesen, J., Renals, S.: Lightly supervised automatic subtitling of weather forecasts. In: Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding, (ASRU), pp. 452-457 (2013).
7. Google. Automatic captions in YouTube. Google Official Blog. http://googleblog.blogspot.com/2009/11/automatic-captions-in-youtube.html (2009).
8. Neto, J., Meinedo , H., Viveiros, M., Cassaca, R., Martins, C., Caseiro, D.: Broadcast news subtitling system in Portuguese. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Las Vegas, USA (2008).
9. Perego, E.: Subtitles and line-breaks: Towards improved readability. In: Chiaro, D., Heiss, C., Bucaria, C. (eds.) Between text and image. Updating research in screen translation, pp. 211–223. Amsterdam, the Netherlands: John Benjamins (2008).
10. Rajendran, D. J., Duchowski, A. T., Orero, P., Martínez, J., Romero-Fresco, P.: Effects of text chunking on subtitling: A quantitative and qualitative examination. In: Perspectives 21, no. 1, pp. 5-21 (2013).
11. AENOR, Spanish technical standards. Standard UNE 153010:2003: Subtitled through teletext. http://www.aenor.es (2003)
12. Díaz-Cintas, J., Orero, P., Remael, A. (eds.): Media for all: subtitling for the deaf, audio description, and sign language (Vol. 30). Rodopi (2007).
13. D’Arcais, F., Giovanni, B.: Syntactic Processing during Reading for comprehension. In: Coltheart, M. (ed.) Attention and Performance II: The Psychology of Reading, London: Lawrence Erlbaum Associates, pp. 619-633 (1987).
14. Coltheart, M. (ed.): What Would We Read Best? In: Coltheart, M. (ed.) Attention and Performance II: The Psychology of Reading, London: Lawrence Erlbaum Associates (1987)
15. Karamitroglou, F.: A Proposed Set of Subtitling Standards in Europe. In: Translation Journal 2(2), pp. 1-15, http://accurapid.com/journal/04stndrd.htm. (1998).
16. Padró, L., Stanilovsky, E.: FreeLing 3.0: Towards Wider Multilinguality. In: Proceedings of the Language Resources and Evaluation Conference (LREC). Istanbul, Turkey (2012).
17. Baldridge, J. The OpenNLP project. http://opennlp.sourceforge.net/ (2005).
18. Petrov, S., Klein, D.: Improved inference for unlexicalized parsing. In: Proceedings of the Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics. Rochester (2007).
19. Klein, D., Manning, C.: Stanford Parser 1.6. http://nlp.stanford.edu/software/lex-parser.shtml (2007).
20. Ezeiza, N., Aduriz, I., Alegria, I., Arriola, J.M., Urizar, R.: Combining Stochastic and Rule-Based Methods for Disambiguation in Agglutinative Languages. In: Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistic. Montreal, Canada (1998).
21. Kenneth, H.: KenLM: Faster and Smaller Language Model Queries. In: Proceedings of the 6th Workshop on Statistical Machine Translation, pp. 187–197 (2011).
22. Kneser, R., Ney, H.: Improved backing-off for m-gram language modeling. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pages 181–184 (1995).
23. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology (2011).
24. Pedregosa, F., et al.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825-2830 (2011).